21M.380

Music and Technology: Algorithmic and Generative Music Systems

Christopher Ariza

Table of Contents

Chapter 1. Meeting 1, Foundations: Algorithmic and Generative Music Systems

1.1. Announcements

- 21M.380: Music Technology: Algorithmic and Generative Music Systems

1.2. Overview

- The last 10 years of algorithmic and generative music systems
- What are algorithmic and generative music systems?
- Two examples
- About this course

1.3. Generative Systems: Definitions

- Machines that make music
- Humans that use or make machines to make music
- Humans that use or make machines to help them make music
- Humans that use or make machines to help them make components of their music

1.4. A New Field of Compositional Research

- Generative music with a computer took many names:
  - Algorithmic composition
  - Computer music
  - Score synthesis
  - Computer-aided (or -assisted) composition
- Computer-aided algorithmic composition (CAAC)
- A new type of generative (rather than reductive) music theory

1.5. Computer-Aided Algorithmic Composition: Definition

- A negative definition
- A CAAC system is software that facilitates the generation of new music by means other than the manipulation of a direct music representation (Ariza 2005b)
- New music: a unique musical variant, not just as copy
- Output may be in the form of any sound or sound parameter data, from a sequence of samples to the notation of a complete composition
- A "direct music representation" refers to a linear, literal, or symbolic representation of complete musical events, such as an event list (a score in Western notation or a MIDI file) or an ordered list of amplitude values (a digital audio file or stream)
- If the representation provided to the user is the same as the output, the representation may reasonably be considered direct.
- Anything that is not a direct representation employs CAAC

1.6. A Wide Range of Interactions and Collaborations

- Machines can be used to create complete structucres
- Machines can be used to create small fragments that are manually integrated
- Machines can be used to create guidelines, contexts, or situations for human music making

1.7. Two Examples

- I: Minuets and Contredances
- II: babelcast

1.8. I: Minuets and Contredances

- Minuet: a French dance in moderate triple meter, popular in aristocratic society from mid 17th century to late 18th century (Grove Music Online)
- Textbook composition method: two or four bar groups, each section being 8 or 16 bars long
- Audio played in class: Bach: Minuet in G, MWV Anh 114

- Audio played in class: Mozart: Minuet in G, K. 1

1.9. I: Minutes and Contredances: Musical Dice Games

- 1757-1812: at least 20 musical dices games published (Kirnberger, CPE Bach, J Haydn, Mozart, others)
- Musical composition game, one of many 18th-century parlor games (Hedges 1978, p. 180)
- A table is used to translate the sum of two dice to appropriate score positions
- Score positions specify complete measure-length segments for each possible phrase position
- German composer Kirnberger published one of the first in 1757

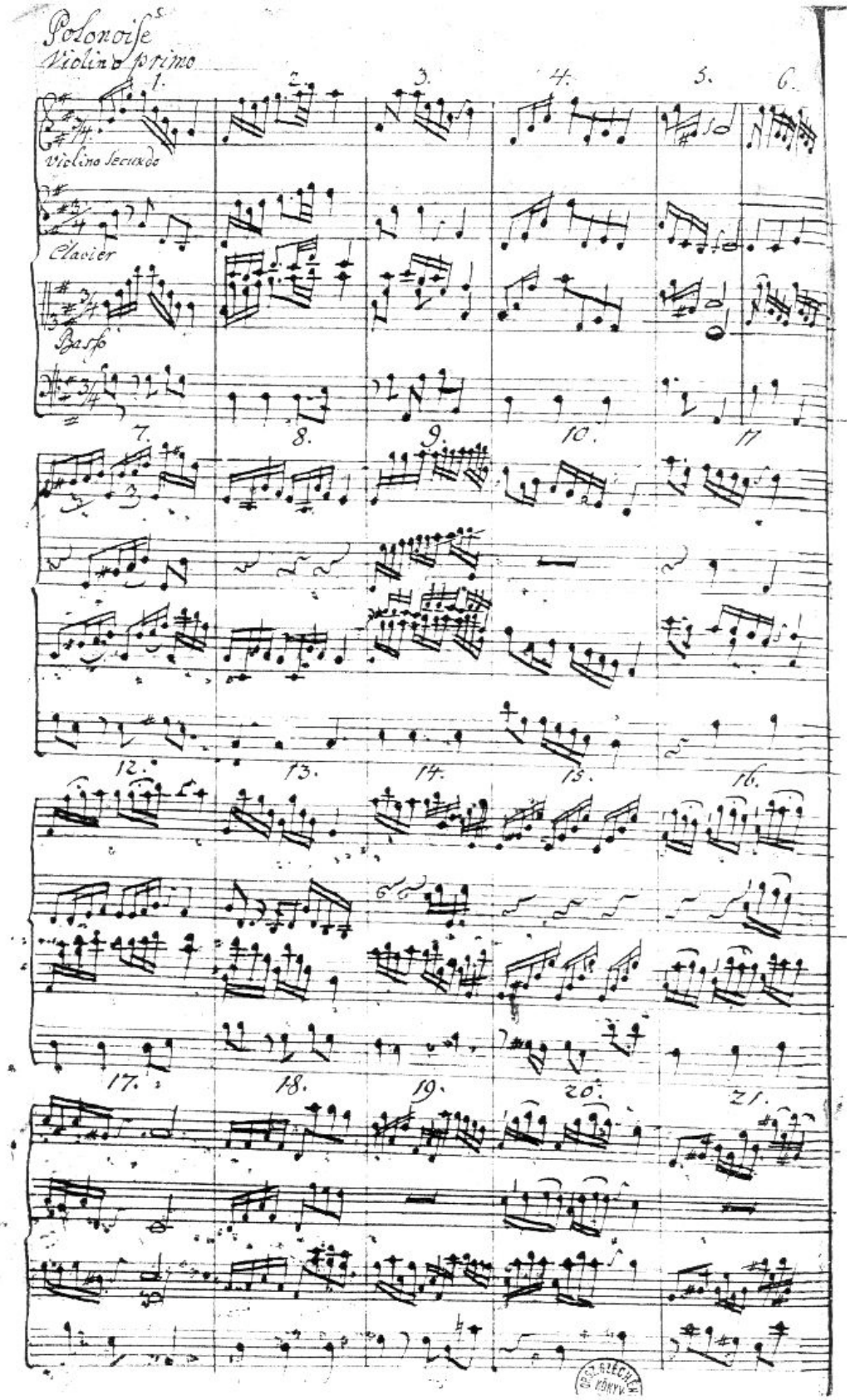
Tabelle der Würffe zu Polonoisen

Zum ersten Theil

Theil seines Würffels	1.	2.	3	4.	5.	6	.	.	.	.	.
Theil seines Würffels	2	3	4	5	6	7	8	9	10	11	12
Der 1 Zug	70	10	42	62	44	72	114	123	131	138	142
Der 2 Zug	34	24	6	8	56	30	112	116	147	151	153
3 Zug	68	50	60	36	40	4	126	137	143	118	146
4 Zug	18	46	2	12	79	28	87	110	113	124	128
5 Zug	32	14	52	16	118	22	89	91	107	141	150
6 Zug	58	26	66	38	54	64	88	98	115	127	134

Zum zweyten Theil.

Theil seines Würffels	1	2	3	4.	5	6	.	.	.	.	.
Theil seines Würffels											
Der 1 Zug	80	20	82	43	78	69	90	129	103	143	152
2 Zug	11	77	2	41	84	62	92	99	140	149	102
3 Zug	59	65	9	45	29	7	86	107	111	97	135
4 Zug	35	5	83	17	75	47	94	122	145	134	148
5 Zug	74	24	67	37	63	19	96	105	133	120	136
6 Zug	10	71	1	49	57	21	85	92	109	102	108
7 Zug	21	15	53	72	51	81	95	106	117	119	130
8 Zug	33	39	25	23	76	55	104	121	125	132	139


Polonoise
Violino primo
Violino secundo
Clavier
Basso

- Numerous versions of *Musikalisches Würfelspiel* attributed to Mozart
- The version attributed to Mozart was first published two years after his death by Juhan Julius Hummel (1793) and includes two similar games: one for Minuets and another for contredances
- Two 8-bar phrases are created from combining 176 pre-composed measures
- The last bar of each phrase always uses the same measure

1.10. I: Minuets and Contredances: The First Computer Implementation

- 1955: David Caplin and Dietrich Prinz write a program to generate and synthesize the Mozart Dice Game for contredances on a Ferranti Mark 1* (MIRACLE) at Shell laboratories in Amsterdam (Ariza 2010)
- Likely the first use of a computer to generate music
- Ferranti Mark 1* (MIRACLE)

Audio sample played in class.

1.11. I: Minutes and Contredances: Motivations and Meanings

- Why do this? How is this possible?
- Is new music being made?
- What meaning, if any, is conveyed?

1.12. II: The babelcast

- An algorithmic, computer generated podcast series (Ariza 2007b)

  Audio RSS URL: (http://www.flexatone.net/babelcast.xml)

  Video RSS URL: (http://www.flexatone.net/babelcast-zoetrope.xml)
- First released 5 August 2005, around one episode a month since
- Created with athenaCL, Python, and Csound
- Distributed in three formats: mp3, (-mosaic) m4a, and (-zoetrope) m4v

1.13. II: The babelcast: Information Abduction and Reduction

- Gather sounds of politicians and political commentators
- Gather images of politicians and political commentators
- Favor primary sources
- Favor massively redundant surplus media: images and sounds that are obtained by many sources

1.14. II: The babelcast: The Process

- Sounds are manually collected with minimal editing
- images are automatically downloaded and then manually filtered
- Around 40 Texture-generating procedures for athenaCL are configured for each episode
  - Some Textures create noises

    - Some Textures process samples
    - Csound instruments use vocoders, granular synthesis methods, and other techniques
- Between 100 and 200 Textures are generated and mixed into a single audio file
- Images are randomly selected, cropped, and zoomed

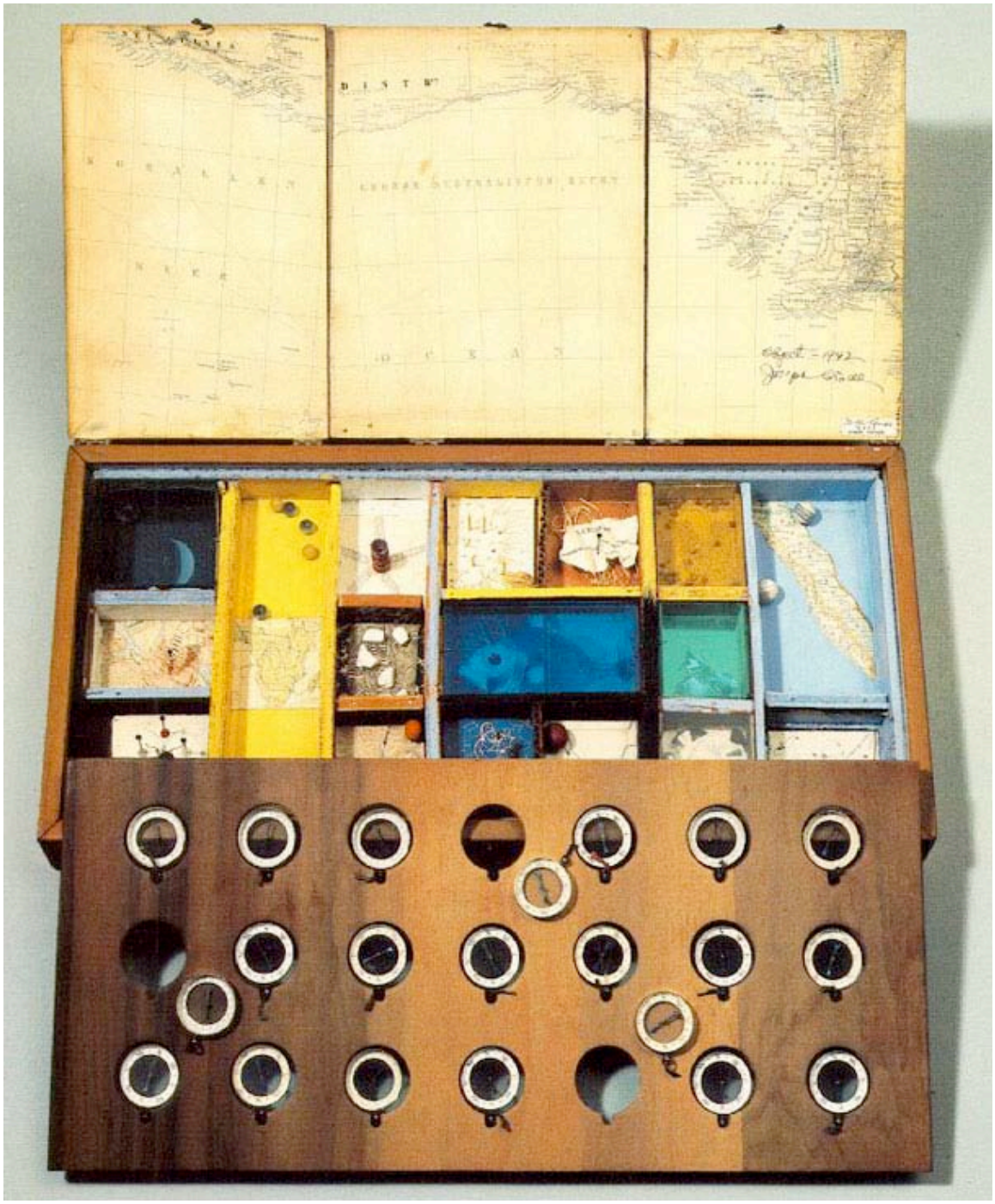
1.15. II: Listening

- babelcast-zoetrope-2009.12.27

(http://www.flexatone.net/video/m4v/babelcast-zoetrope-2009.12.27.m4v)

1.16. II: The babelcast: Precedents

- 1989: Umberto Ecco, *The Open Work*
    - Leaving parts of a work to chance
    - Works that "reject the definitive, concluded message and multiply the formal possibilities of the distribution of their elements" (Eco 1989, p. 3).
- 1986: William Gibson, *Count Zero*
    - Artificial intelligence that sends randomly constructed human junk, found in space, back down to earth, which is assumed to be forged works of artists Joseph Cornell
    - American "assemblage" artist Joseph Cornell (1903-1972)
    - Cornell: *Object (Roses des Vents)* (1942-53)



1.17. II: The babelcast: Motivations and Meanings

- Why do this?
- What meaning, if any, is conveyed?

1.18. 21M.380: Objectives

- To gain a critical understanding of the history, techniques, and designs of algorithmic and generative music systems
- To develop musical creativity and expression in the use and design of algorithmic and generative music systems
- To critically evaluate claims of aesthetic and technological advancement, quality, and promise

1.19. 21M.380: Areas of Focus

- History: Mechanical Musical Automata, Serialism, Phasing, Gottfried Michael Koenig, Lejaren Hiller, Iannis Xenakis
- Approaches: Distributions and Stochastics, Probability and Markov Chains, Cellular Automata, Genetic Algorithms, Grammars and L-Systems, Agents and Ecological Models, Expert Systems and Style Emulation, Non-Standard Synthesis, Granular and Concatenative Synthesis, Mapping, Sonification, and Data Bending
- Workshops and Discussion

1.20. 21M.380: Prerequisites

- None but curiosity, willingness to experiment
- Programming in Python or other languages useful, but not required
- Experience with digital audio and DAW software desirable, but not required

1.21. 21M.380: Course Meetings and Materials

- Syllabus:
- Two types of meetings
  - Topic meetings: focused on material in readings, listening, and themes, combining lecture, discussion, demonstration, and listening
  - Workshop meetings: focus on discussion of projects and techniques, hands-on experimentation
  - If possible, bring laptops to all class meetings
- Software: core tools
  - athenaCL

    - Python
    - Csound
    - SuperCollider
    - PD
    - DAWs and virtual instruments
- Lecture notes

1.22. 21M.380: Assignments: Reading

- Numerous carefully selected readings

Ames, C. 1987. "Automated Composition in Retrospect: 1956-1986." *Leonardo* 20(2): 169-185.

Ames, C. 1992. "A Catalog of Sequence Generators: Accounting for Proximity, Pattern, Exclusion, Balance and/or Randomness." *Leonardo Music Journal* 2(1): 55-72.

Ames, C. 1991. "A Catalog of Statistical Distributions: Techniques for Transforming Random, Determinate and Chaotic Sequences." *Leonardo Music Journal* 1(1): 55-70.

Ames, C. 1989. "The Markov Process as a Compositional Model: A Survey and Tutorial." *Leonardo* 22(2): 175-187.

Ariza, C. 2007a. "Automata Bending: Applications of Dynamic Mutation and Dynamic Rules in Modular One-Dimensional Cellular Automata." *Computer Music Journal* 31(1): 29-49. Internet: http://www.mitpressjournals.org/doi/abs/10.1162/comj.2007.31.1.29.

Ariza, C. 2006. "Beyond the Transition Matrix: A Language-Independent, String-Based Input Notation for Incomplete, Multiple-Order, Static Markov Transition Values." Internet: http://www.flexatone.net/docs/btmimosmtv.pdf.

Ariza, C. 2009a. "The Interrogator as Critic: The Turing Test and the Evaluation of Generative Music Systems." *Computer Music Journal* 33(2): 48-70. Internet: http://www.mitpressjournals.org/doi/abs/10.1162/comj.2009.33.2.48.

Ariza, C. 2005b. "Navigating the Landscape of Computer-Aided Algorithmic Composition Systems: A Definition, Seven Descriptors, and a Lexicon of Systems and Research." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 765-772. Internet: http://www.flexatone.net/docs/nlcaacs.pdf.

Ariza, C. 2005c. "The Xenakis Sieve as Object: A New Model and a Complete Implementation." *Computer Music Journal* 29(2): 40-60. Internet: http://www.mitpressjournals.org/doi/abs/10.1162/0148926054094396.

Ben-Tal, O. and J. Berger. 2004. "Creative Aspects of Sonification." *Leonardo Music Journal* 37(3): 229-232.

Berg, P. 2009. "Composing Sound Structures with Rules." *Contemporary Music Review* 28(1): 75-87.

Biles, J. A. 2003. "GenJam in Perspective: A Tentative Taxonomy for GA Music and Art Systems." *Leonardo* 36(1): 43-45.

Cope, D. 1992. "Computer Modeling of Musical Intelligence in EMI." *Computer Music Journal* 16(2): 69-83.

Ebcioglu, K. 1988. "An Expert System for Harmonizing Four-part Chorales." *Computer Music Journal* 12(3): 43-51.

Hiller, L. and L. Isaacson. 1958. "Musical Composition with a High-Speed Digital Computer." *Journal of the Audio Engineering Society* 6(3): 154-160.

Hoffman, P. 2000. "A New GENDYN Program." *Computer Music Journal* 24(2): 31-38.

Koenig, G. M. 1971. "The Use of Computer Programs in Creating Music." In *Music and Technology (Proceedings of the Stockholm Meeting organized by UNESCO)*. Paris: La Revue Musicale. 93-115. Internet: http://www.koenigproject.nl/Computer_in_Creating_Music.pdf.

Koenig, G. M. 1983. "Aesthetic Integration of Computer-Composed Scores." *Computer Music Journal* 7(4): 27-32.

Magnus, C. 2004. "Evolving electroacoustic music: the application of genetic algorithms to time-domain waveforms." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 173-176.

Marino, G. and M. Serra, J. Raczinski. 1993. "The UPIC System: Origins and Innovations." *Perspectives of New Music* 31(1): 258-269.

Mason, S. and M. Saffle. 1994. "L-Systems, Melodies and Musical Structure." *Leonardo Music Journal* 4: 31-38.

Miranda, E. R. 2003. "On the Music of Emergent Behavior: What Can Evolutionary Computation Bring to the Musician?." *Leonardo* 36(1): 55-59.

Riskin, J. 2003. "The Defecating Duck, or, the Ambiguous Origins of Artificial Life." *Critical Inquiry* 29(4): 599-633.

Roads, C. 1988. "Introduction to Granular Synthesis." *Computer Music Journal* 12(2): 11-13.

Rowe, R. 1992. "Machine Listening and Composing with Cypher." *Computer Music Journal* 16(1): 43-63.

Serra, M. 1993. "Stochastic Composition and Stochastic Timbre: GENDY3 by Iannis Xenakis." *Perspectives of New Music* 31(1): 236-257.

Soldier, D. 2002. "Eine Kleine Naughtmusik: How Nefarious Nonartists Cleverly Imitate Music." *Leonardo Music Journal* 12: 53-58.

Sturm, B. L. 2006. "Adaptive Concatenative Sound Synthesis and Its Application to Micromontage Composition." *Computer Music Journal* 30(4): 46-66.

Voss, R. F. and J. Clarke. 1978. "1/f Noise in Music: Music from 1/f Noise." *Journal of the Acoustical Society of America* 63(1): 258-263.

Xenakis, I. 1971. "Free stochastic Music." In *Cybernetics, art and ideas*. J. Reichardt, ed. Greenwich: New York Graphic Society. 124-142.

Xenakis, I. 1987. "Xenakis on Xenakis." *Perspectives of New Music* 25(1-2): 16-63.

1.23. 21M.380: Assignments: Listening

- Reading notation and scores not required
- Take notes when you listen
- What to listen for: duration, instrumentation, method of production, recording or performance context, notable sonic events, form, temporal design and proportions, aesthetic or historical contexts, and/or critical and subjective responses

1.24. 21M.380: Assignments: Discussion Leaders

- Students are assigned to cover each reading and listening assignments for each class
- Must be available to lead discussion, answer questions, and provide a resource to class
- Must post minimal notes in the class website forum: Reading and Listening Notes

1.25. 21M.380: Assignments: Musical Design Report

- An original sonic sketch or musical work, lasting from two to five minutes, realized in notation, MIDI, digital audio, or code, and based on approaches, techniques, and/or models presented for each assignment
- Includes a very short written report describing approaches and design
- A group of 3 to 4 students will be selected to present their projects to the class during Workshop sessions
- Three spaced evenly throughout the semester

1.26. 21M.380: Assignments: Sonic System Project and Presentation

- An original sonic system that functions as either a generative instrument with or without a performance interface or as a static or dynamic musical work employing techniques and/or tools of algorithmic composition.
- May explore any software or hardware system or interface; can extend class examples or produce completely original works
- Includes a short written report describing approaches and design
- Draft workshop meeting: 27 April
- Final presentations: 11 and 13 May

1.27. 21M.380: Assignments: Submission

- All assignments are submitted digitally via email attachment (or as Forum posts)
- All assignments, except as noted, are due at 11:59:59 PM on due date
- Late within 1 week: 20% reduction; no assignments accepted after 1 week

1.28. 21M.380: Attendance

- Mandatory and essential
- More than one unexcused absence incurs a 3% grade reduction

1.29. 21M.380: Exams and Quizzes

- Quizzes will be announced, and frequent

- All short written answers
- Quizzes will be based on reading, listening, and course content
- No final exam

1.30. 21M.380: Grading

- Reading and Listening Discussion Leader: 20%
- Musical Design Report (3): 30%
- Sonic System Project and Presentation: 20%
- Sonic System Project Draft: 5%
- Quizzes: 15%
- Participation: 10%

1.31. 21M.380: Additional Policies

- Read entire syllabus
- Common courtesies
- Computers in class
- Academic integrity

1.32. 21M.380: Contact

- Always feel free to contact me with any problem or concern with this class

1.33. Us

- Backgrounds, experiences, goals

1.34. For Next Class

- Download and read entire syllabus
- Respond to my email questionnaire
- Bring computers

[pp. 17-19 deleted from these notes, due to privacy considerations]

Chapter 2. Meeting 2, Foundations: Musical Parameters, Mappings, and Tools

2.1. Announcements

- If you have not downloaded and installed Python and PD-Extended, please do so now
- Download: most recent athenaCL

  http://code.google.com/p/athenacl

2.2. Overview

- Events
- Parameters
- Containers
- Instruments
- Generative software tools
- athenaCL and Python
- Digital Audio Workstations

2.3. Musical Events

- The event is the fundamental unit of music
- An event can be single sample lasting 0.0000227 seconds
- An event can be a note
- An event can be a continuous sound encompassing a complete work
- The minimum definition of an event is a start and end time

2.4. Events and Parameters

- An event can be described as with one or more parameters
- Parameters may be duration, pitch, amplitude, or any other collection of specifiers
- Parameters may be coordinated or independent
- Human musical production often coordinates parameters
- Independent musical paramerers can make interest musical structures
- The parameterization of musical events has been critical to the development of modern music

2.5. Event Lists

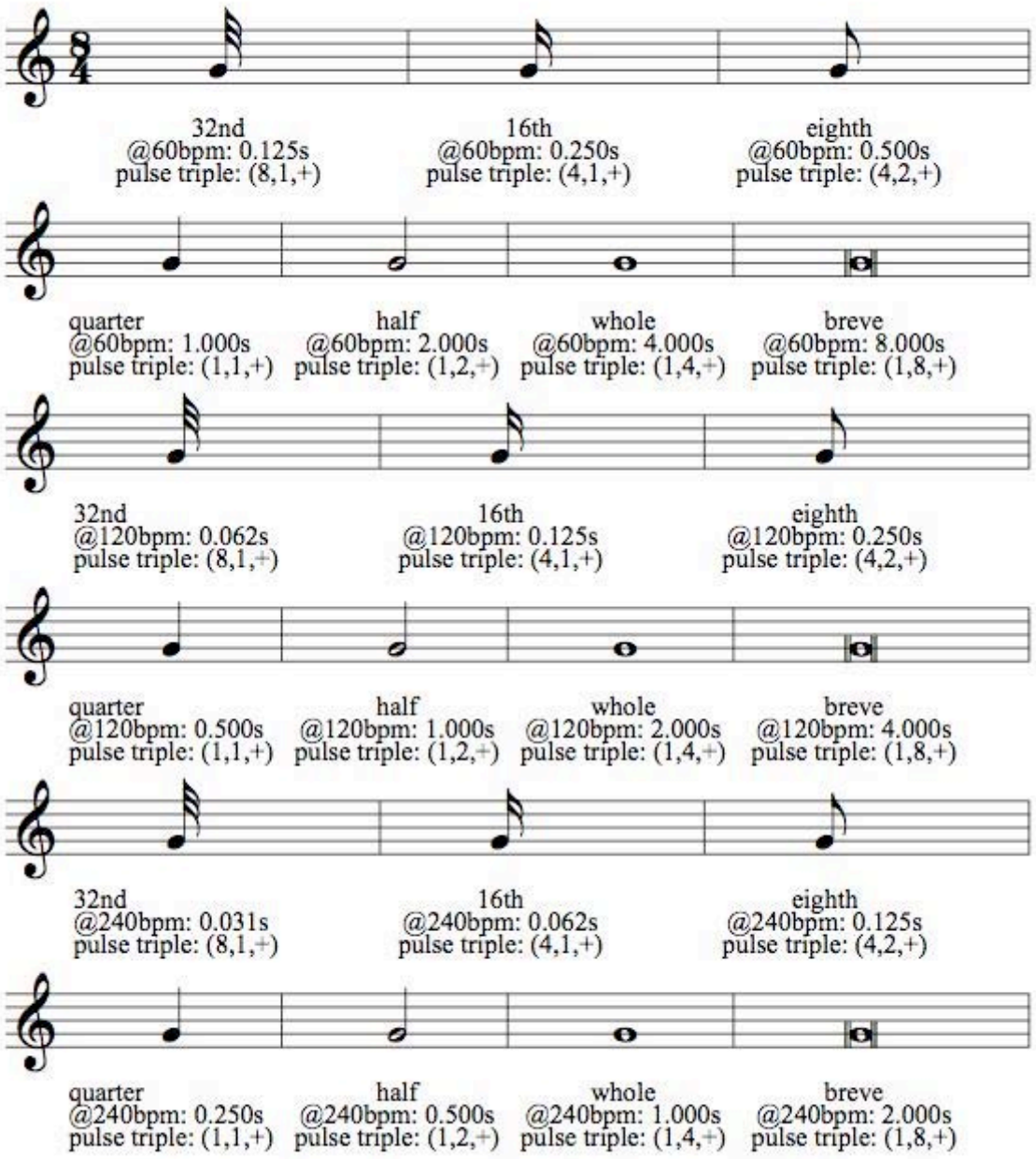
- Events, defined by an array of parameters, can be collected in a list
- Musical data is stored in various arrangements of event lists

2.6. Fundamental Musical Parameters

- Duration and rhythm
- Frequency and pitch
- Amplitude and dynamics

2.7. Parameters: Duration and Rhythm

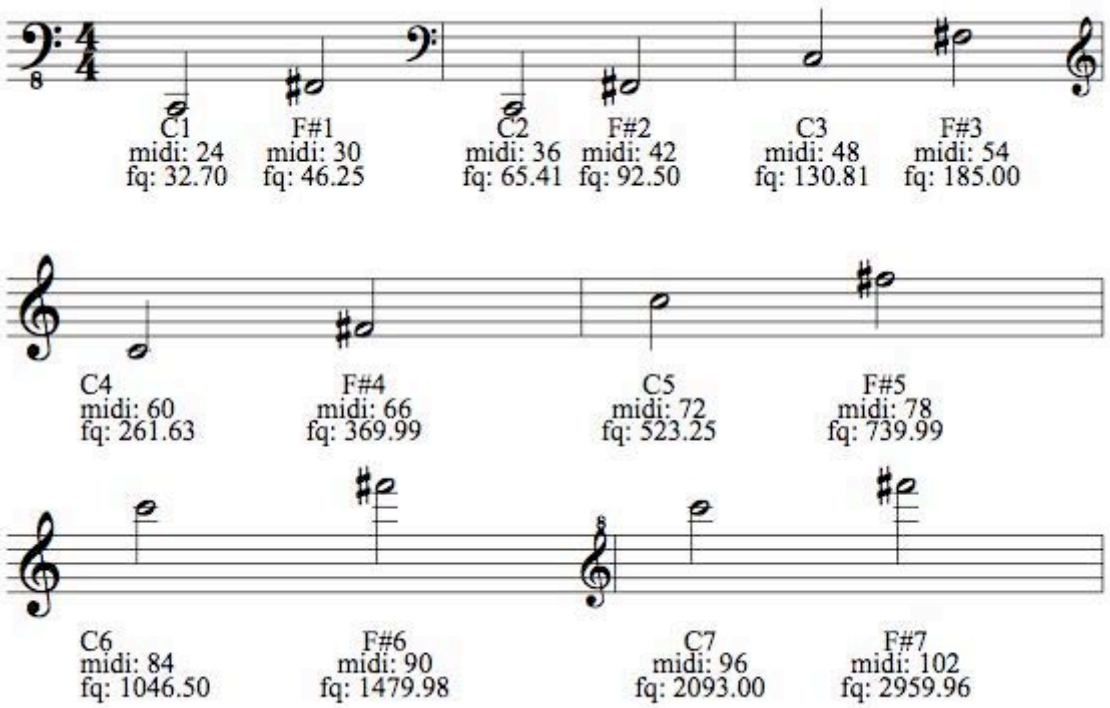
- Can be measured in absolute or relative values
- Absolute values: seconds, milliseconds
- Relative values
  - Notation: quarter, sixteenth, whole
  - Pulse triples: (divisor, multiplier, accent)
- Relative values proportional to a beat rate (tempo)
- Tempi are often thought of in beats per minute (BPM)
- A range of durations at different tempi [py/demo/parameterDuration.py]



2.8. Parameters: Frequency and Pitch

- Pitch is a human interpretation of frequency
- Pitch asserts the octave as referential unit of equivalence
- An octave is 12 half steps, 8 diatonic steps (white notes on the piano), and a 2:1 frequency ratio
- Numerous other distances between pitches (intervals) have names: fifths, thirds, 13ths, quarter tones

- Pitch names can carry octave designation, where C4 is middle C
- MIDI pitch values place C4 at 60, use 1 as a half step, and range from 0 to 127
- athenaCL pitch space values place C4 at 0 and use 1 as a half step
- A range of fundamental pitches [py/demo/parameterPitch.py]

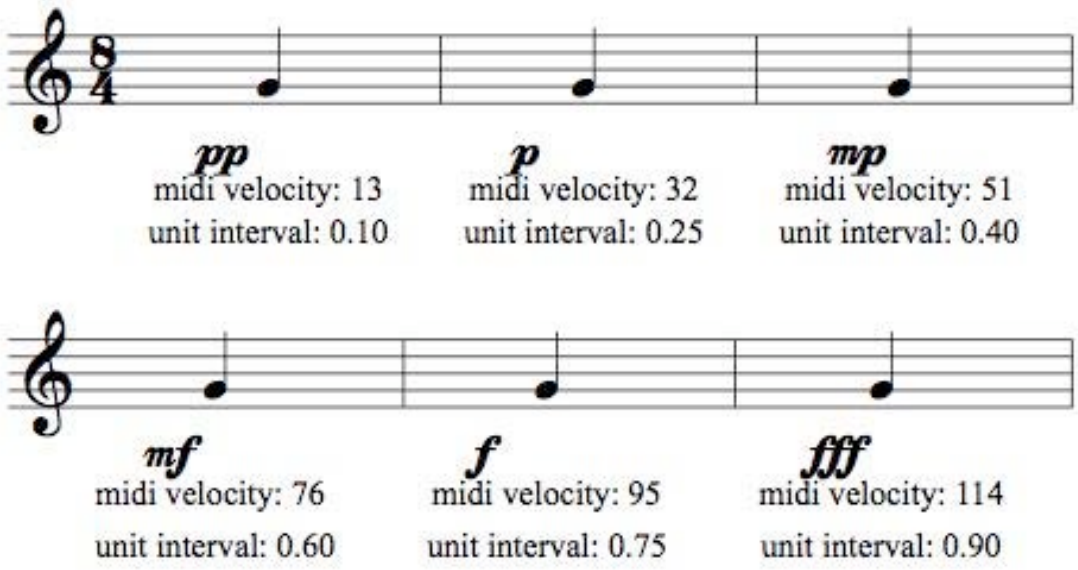


- The (ideal) audible frequency range: 20 Hz (MIDI 16, E0) to 20000 Hz (MIDI 135, D#10)
- Top three octaves (from 3-6k, 6-12k, 12-24k) contain spectral frequencies

2.9. Parameters: Amplitude and Dynamics

- Bits: discrete digital audio amplitude levels
- dB SPL: acoustic power
- dBv: voltage amplitude
- Unit interval spacings: between 0 and 1
- Notation: from *ppp* to *fff*

- MIDI velocity values from 0 to 127
- A range of amplitude levels [py/demo/parameterAmplitude.py]



2.10. Storing Event Lists in Containers

- Events can be streamed in real-time or stored in containers
- Western notation (scores, MusicXML)
- Musical Insturment Digital Interface (MIDI)
- Open sound control (OSC)
- Digital audio files

2.11. Containers: Western Notation

- Events are organized around notes that specify pitch and duration
- Parameter values are limited (mostly) to symbols
- Parameters are isolated for instruments by staves
- Parallel staves express simultaneous events
- Timbral specification relies mostly on instrument assignments

- MusicXML offers a standard for encoding notation
- Software permits opening, editing, and playing MusicXML files
  - Finale and Sibelius
  - Finale reader

    http://www.finalemusic.com/Reader

2.12. Containers: MIDI

- A binary representation of musical parameters
- Parameter values are often 7 bit, or 128 discrete values
- Parameters are isolated by numerical tags, called channels
- Timbral specification relies mostly on instrument assignments (programs)
- Software permits performing MIDI files
  - QuickTime and Windows Media Player
  - Virtual instruments

2.13. Containers: OSC

- A hierarchical representation of musical parameters
- Parameter values can be numbers or strings
- Parameters are organized hierarchically with URL-like syntax
- Timbral specification relies mostly on receiving device
- Sending and receiving OSC data
  - Hardware controllers
  - Software controllers

2.14. Containers: Digital Audio

- A micro mono-parameter representation
- Store amplitude values within a dynamic range taken at a sampling rate

- Signals can be mixed or stored in isolated channels
- Digital audio is a timbral specification
- Software permits playing and editing digital audio
  - QuickTime and Windows Media Player
  - Audacity

    http://audacity.sourceforge.net
  - Digital Audio Workstations

2.15. Synthesizers, Samplers, and Virtual Instruments

- Acoustic instruments translate parameters into acoustic sound
- Electronic instruments synthesize tones with oscillators or stored samples
- Digital electronic instruments are built by combining basic software components
- Virtual instruments are software synthesizers or samplers that respond to MIDI or OSC parameters

2.16. Digital Synthesizers

- Built from combing fundamental signal generators and processors (unit generators or Ugens)
- Can be designed to accept any number of initial event parameters
- Can be designed to accept dynamic parameters over the course of an event

2.17. Digital Synthesis Languages: Csound

- Developed in part from the first synthesis language Music 1 in 1957 (Roads 1980)
- Extended and ported by Barry Vercoe at MIT (1986)
- A huge library of processors and instrument models (Boulanger 2000)
- A low-level language for defining instruments
- A flat list of data for event lists

2.18. Digital Synthesis Languages: PureData

- Over 20 years of development in synthesis, sampling, and a visual programming envrionment
- Numerous related alternatives: Max/MSP, jMax, Open Sound World (OSW)
- Developed by Miller Puckette, creator of the first Max (Puckette 1985, 1988, 1997, 2002)

2.19. Digital Synthesis Languages: SuperCollider

- An extension of Csound archetypes into a modern language and network archetype
- First released in 1996 by James McCartney (McCartney 1996; McCartney 1998)
- A complete object-oriented language: create objects, manipulate, and reuse code
- A server-based architecture: SynthDefs live on a server and send and receive messages and signals
- Designed for real-time performance and experimentation

2.20. Virtual Instruments

- Software plug-ins that can receive MIDI or OSC messages
- Distributed as VST, AU, or other plug-in formats
- Can employ any internal software and synthesis model

2.21. Algorithmic Composition and Generative Music Systems

- May be built within a synthesis language
- May be stand-alone systems
- Numerous systems support multiple output formats from a single interface
  - athenaCL

    http://code.google.com/p/athenacl
  - AC Toolbox: Lisp based Macintosh application/environment

    http://www.koncon.nl/downloads/ACToolbox/
  - Open Music: Lisp based visual programming language

http://recherche.ircam.fr/equipes/repmus/OpenMusic/

- Common Music: Lisp based programming language

  http://commonmusic.sourceforge.net

2.22. A Brief History of athenaCL

- Started as a way of automating the production of Csound scores in 2001
- Originally attempted to integrate a variety of post-tonal music theory tools
- Gradually became a more general tool for composition
- A way to test and deploy modular approaches to generating music parameters and structures
- Support for output in MIDI, SuperCollider, and other formats incrementally added
- Version 2 strips away post-tonal music theory tools, focuses on compositional tasks
- Present alpha releases may have bugs: please report any problems to me immediately

2.23. Installing and Running athenaCL

- Download the most-recent version
  - A distribution from Google Code

    http://code.google.com/p/athenacl
  - Via SVN command-line argument:

    ```
    svn checkout http://athenacl.googlecode.com/svn/trunk/ athenacl-read-only
    ```

- Install in Python's site packages
  - Windows: run athenaCL.exe installer
  - Others: extract athenaCL.tar.gz

    With terminal, cd to athenaCL directory

    Enter: python setup.py install

    If permissions error, try: sudo python setup.py install
- Start Python

    - Windows: run python.exe or IDLE.py
    - Others: open terminal, enter: python
- Start athenaCL
    - From within Python, enter: from athenaCL import athenacl
    - Others: open terminal, enter: python

      Enter: from athenaCL import athenacl

2.24. Running athenaCL Without Installing

- Download athenaCL (as above)
- Launch the file athenaCL/athenacl.py with Python

2.25. athenaCL: System Overview

- Create and edit Textures (TextureInstances) and Paths (PathInstances)
- Paths are static pitch collections
- Textures are dynamic variable parameter event list generators
- TextureModules define various approaches to create Textures
- ParameterObjects are used to configure and generate parameters within Textures
- EventModes define orchestras of instruments and available output formats
- EventOutputs are output formats, some available with all EventModes, others available from only one
- EventLists can be created, rendered, and heard

2.26. Interactive athenaCL Commands

- athenaCL as an interactive command line program
- Commands can be provided with space delimited arguments, or the user can be prompted for all necessary arguments
- Acronyms are always accepted for arguments

- cmd: view all commands
- ?: get help for any command
- EMo: select EventMode midiPercussion
- EMi: list available instruments
- TIn: create a new TextureInstance (provide name and instrument number)
- ELn: create a new EventList
- ELh: hear (or open) a new EventList
- Commands with full arguments and sample output

```
pi{}ti{} :: emo mp
EventMode mode set to: midiPercussion.

pi{}ti{} :: tin a 50
TI a created.

pi{auto-highTom}ti{a} :: eln
command.py: temporary file: /Volumes/xdisc/_scratch/ath2010.02.04.09.45.48.xml
      EventList ath2010.02.04.09.45.48 complete:
/Volumes/xdisc/_scratch/ath2010.02.04.09.45.48.mid
/Volumes/xdisc/_scratch/ath2010.02.04.09.45.48.xml

pi{auto-highTom}ti{a} :: elh
EventList hear initiated: /volumes/xdisc/_scratch/ath2010.02.04.09.48.11.mid
```

- Setting the scratch directory to "/Volumes/xdisc/_scratch"

```
pi{}ti{} :: apdir x /Volumes/xdisc/_scratch
user fpScratchDir directory set to /volumes/xdisc/_scratch.
```

- Editing the Texture's temp with a WaveSine generator (space divisions matter)

```
pi{auto-highTom}ti{a} :: tie b ws,t,10,0,40,400
TI a: parameter bpm updated.

pi{auto-highTom}ti{a} :: eln; elh
```

2.27. Automating athenaCL Commands with Python

- athenaCL command can be scripted and controlled in Python script
- Permits reuse and extensions
- Must create an athenaCL Interpreter object and send string commands
- Creating a Python script file
  - Windows: use IDLE.py or another text editor

- Others: use emacs, vi, or other text editor
- Mac: use TextWranger (free)

  http://www.barebones.com/products/TextWrangler

- Automating the production of one Texture: create file 02a.py and run with python [02a.py]

```
from athenaCL.libATH import athenaObj

ath = athenaObj.Interpreter()

ath.cmd('emo mp')
ath.cmd('tin a 45')
ath.cmd('tie b ws,t,10,0,40,400')
ath.cmd('eln')
ath.cmd('elh')
```

- If Python cannot find the athenaCL directory (because you were not able to do an install) you must provide to python the file path to the directory containing athenaCL

```
import sys
sys.path.append("/path/to/dir/that/contains/athenacl")

from athenaCL.libATH import athenaObj
ath = athenaObj.Interpreter()

ath.cmd('emo mp')
ath.cmd('tin a 45')
ath.cmd('tie b ws,t,10,0,40,400')
ath.cmd('eln')
ath.cmd('elh')
```

- Automating the production of three Textures [02b.py]

```
from athenaCL.libATH import athenaObj
import random

ath = athenaObj.Interpreter()

ath.cmd('emo mp')
for x in [45, 51, 75]:
    ath.cmd('tin t%s %s' % (x, x))
    ath.cmd('tie t %s,%s' % (random.choice(range(0,10)),
                             random.choice(range(20,30))))
    ath.cmd("tie b ws,t,10,0,40,400")
ath.cmd('eln')
ath.cmd('elh')
```

- If you have trouble running a Python script on Windows, visit:

  http://www.python.org/doc/faq/windows/

Chapter 3. Meeting 3, Approaches: Distributions and Stochastics

3.1. Announcements

- Download: most recent athenaCL

  http://code.google.com/p/athenacl

3.2. Reading: Ames: A Catalog of Statistical Distributions

- Ames, C. 1991. "A Catalog of Statistical Distributions: Techniques for Transforming Random, Determinate and Chaotic Sequences." *Leonardo Music Journal* 1(1): 55-70.
- What does Ames mean by balance, and that there can be a balance that is not fair?
- What is meant by a weight? Why is this term preferable to alternatives?
- The use of statistics here might be considered outside of the discipline of statistics: why?
- Which musical parameters are better suited for discrete values? Which for continuous values?
- Are any distributions dependent on past occurrences?
- Why might the Law of Large Numbers make working with distributions difficult in musical contexts?
- In terms of the distribution output, what are time domain and frequency domain graphs?
- What is the relationship between the Poisson distribution and the Exponential distribution?
- Ames notes that, when working with some distributions, values may have to be discarded: why? What does this say about working with distributions?

3.3. ParameterObjects

- Reusable value selectors and generators

- Can be created and controlled with strings of comma-separated lists
- Values in ParameterObjects can be strings (without quotes), numbers, or lists (delimited by parenthesis or brackets)
- In some cases ParameterObjects, enclosed as a list, can be used inside of other ParameterObjects to generate values
- Three types of ParameterObjects
  - Generator: produce values based on arguments alone
  - Rhythm: specialized for rhythm creation
  - Filter: specialized for transforming values produced from a Texture
- Complete documentation for ParameterObjects, and samples, can be found here:

  http://www.flexatone.net/athenaDocs/www/ax03.htm
- ParameterObject names and string values can always be provided with acronyms
- Trailing arguments, when not provided, are automatically supplied

3.4. ParameterObjects: Viewing Arguments and Output

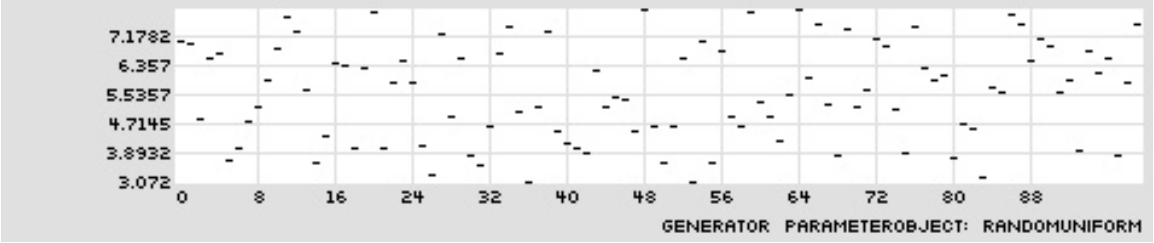
- TPls: view a list of all available ParameterObjects
- TPv: vie detailed documentation for one or more ParameterObjects

```
pi{}ti{} :: tpv ru
Generator ParameterObject
{name,documentation}
RandomUniform       randomUniform, min, max
                    Description: Provides random numbers between 0 and 1 within an
uniform distribution.
                    This value is scaled within the range designated by min and max;
min and max may be
                    specified with ParameterObjects. Note: values are evenly
distributed between min and
                    max. Arguments: (1) name, (2) min, (3) max
```

- TPmap: create a graphical output providing a number of values and a ParameterObject name

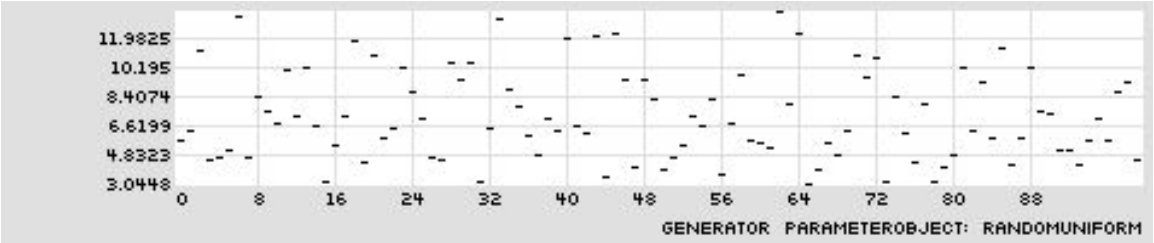
  Note that, when providing arguments from the command-line, spaces cannot be used between ParameterObject arguments

```
pi{}ti{} :: tpmap 100 ru,3,8
randomUniform, (constant, 3), (constant, 8)
TPmap display complete.
```



- With a nested ParameterObject for the maximum value

```
pi{}ti{} :: tpmap 100 ru,3,(ru,8,15)
randomUniform, (constant, 3), (randomUniform, (constant, 8), (constant, 15))
TPmap display complete.
```



3.5. Configuring Graphical Outputs in athenaCL

- athenaCL supports numerous types of graphical outputs, some with various dependencies
- Output formats:
  - JPG, PNG: requires working installation of the Python Imaging Library (PIL)

    Windows: http://www.pythonware.com/products/pil

    Others: not so easy for Python 2.6 (easier for Python 2.5)
  - TK: uses the TK GUI system that ships with Python

    Works for full installs of Python 2.6 on Windows, Mac, Others
  - EPS: works on all Pythons on all platforms
- APgfx: set graphical output preferences

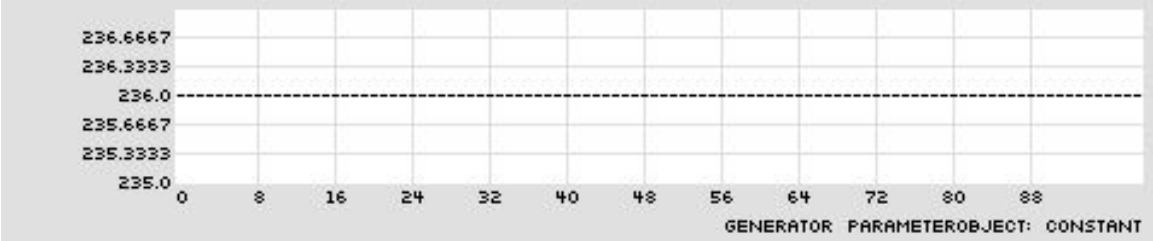
```
pi{}ti{} :: apgfx
active graphics format: png.
select text, eps, tk, jpg, png. (t, e, k, j, or p): p
graphics format changed to png.
```

- Use APea to set the imageViewer and psViewer applications if not already set properly

3.6. The Constant ParameterObject

- The most simple ParameterObject

```
pi{}ti{} :: tpv constant
Generator ParameterObject
{name,documentation}
Constant            constant, value
                    Description: Return a constant string or numeric value.
Arguments: (1) name, (2)
                    value
```



3.7. Continuous and Discrete Stochastic Distributions as ParameterObjects

- Discrete
  - BasketGen
  - Continuous POs put through the Quantize PO or other POs
- Continuous
  - RandomUniform
  - RandomGauss
  - RandomBeta
  - RandomExponential and RandomInverseExponential
  - Many others...

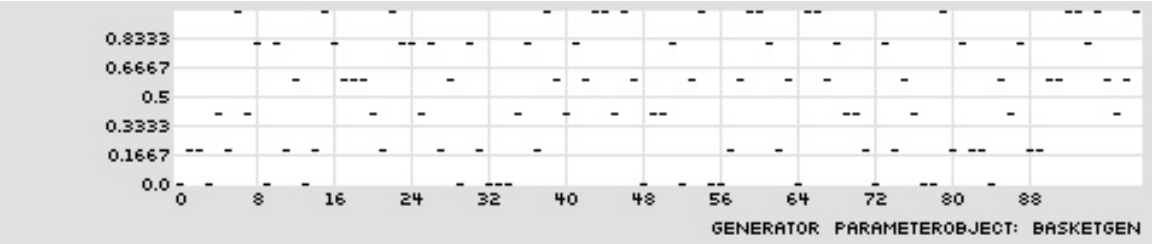
3.8. Discrete Stochastic Distributions as ParameterObjects

- BasketGen: the ball and urn (or basket) paradigm
- Documentation with TPv

```
:: tpv bg
Generator ParameterObject
{name,documentation}
BasketGen           basketGen, selectionString, valueList
                    Description: Chooses values from a user-supplied list
                    (valueList). Values can be strings or numbers. Values are
                    chosen from this list using the selector specified by the
                    selectionString argument. Arguments: (1) name, (2)
                    selectionString {'randomChoice', 'randomWalk',
                    'randomPermutate', 'orderedCyclic',
                    'orderedCyclicRetrograde', 'orderedOscillate'}, (3)
                    valueList
```

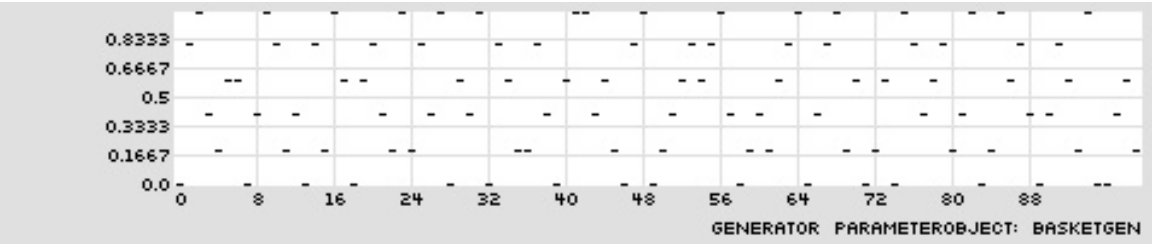
- Selection methods
  - randomChoice: random selection with replacement

```
pi{}ti{} :: tpmap 100 bg,rc,(0,.2,.4,.6,.8,1)
basketGen, randomChoice, (0,0.2,0.4,0.6,0.8,1)
TPmap display complete.
```



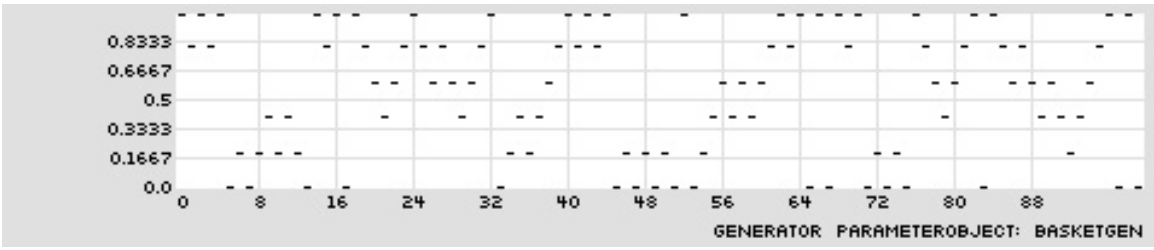
  - randomPermutate: random selection without replacement

```
pi{}ti{} :: tpmap 100 bg,rp,(0,.2,.4,.6,.8,1)
basketGen, randomPermutate, (0,0.2,0.4,0.6,0.8,1)
TPmap display complete.
```



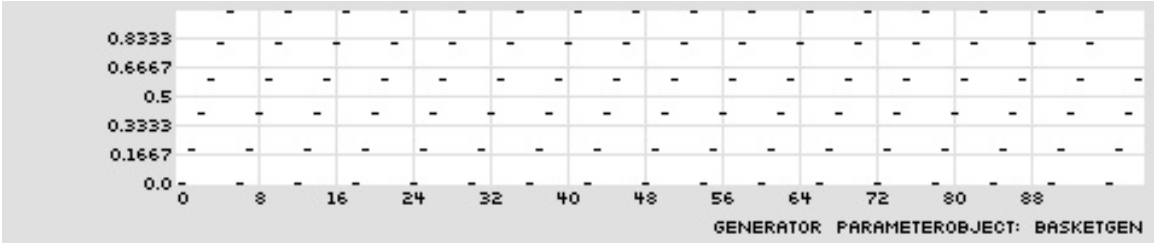
  - randomWalk: random up/down movement along order of list, with wrapping

```
pi{}ti{} :: tpmap 100 bg,rw,(0,.2,.4,.6,.8,1)
basketGen, randomChoice, (0,0.2,0.4,0.6,0.8,1)
TPmap display complete.
```



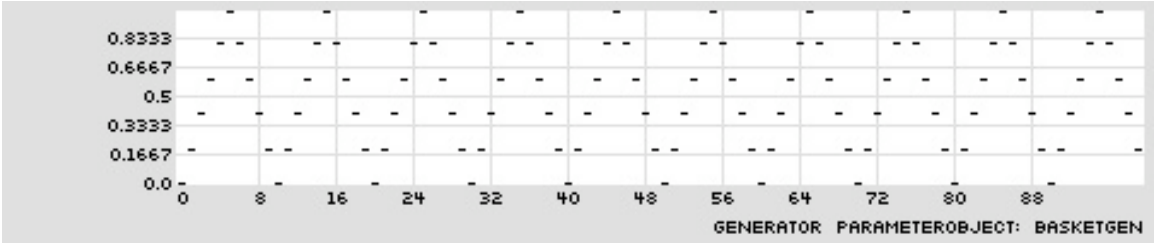
- orderedCyclic: looping

```
pi{}ti{} :: tpmap 100 bg,oc,(0,.2,.4,.6,.8,1)
basketGen, orderedCyclic, (0,0.2,0.4,0.6,0.8,1)
TPmap display complete.
```



- orderedOscillate: oscillating

```
pi{}ti{} :: tpmap 100 bg,oo,(0,.2,.4,.6,.8,1)
basketGen, orderedOscillate, (0,0.2,0.4,0.6,0.8,1)
TPmap display complete.
```



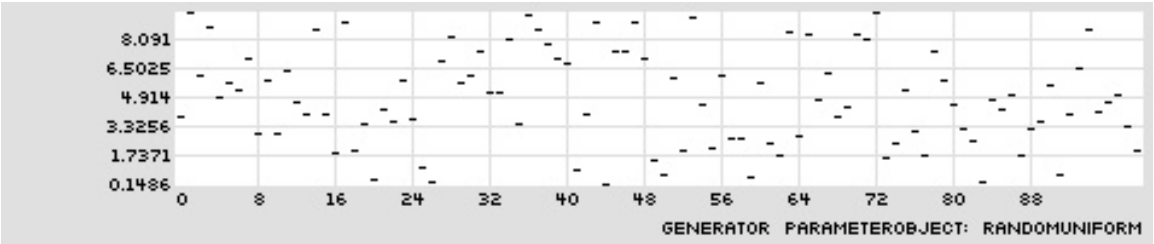
- By configuring the values drawn from, discrete uniform, Bernoulli, and binomial distributions can be modeled

3.9. Continuous Stochastic Distributions as ParameterObjects

- RandomUniform: continuous uniform distribution

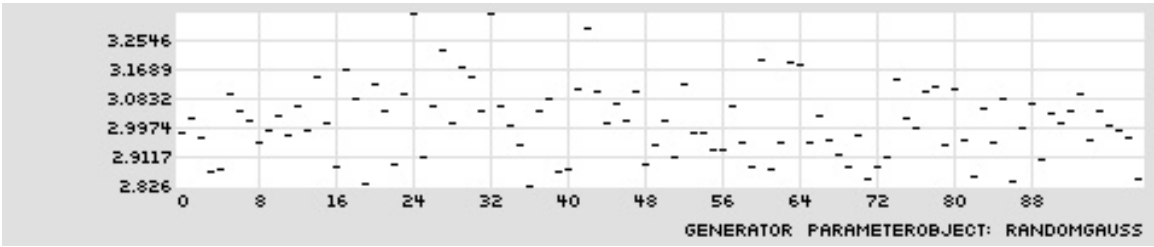
  scaled between 0 and 10

```
pi{}ti{} :: tpmap 100 ru,0,10
randomUniform, (constant, 0), (constant, 10)
TPmap display complete.
```



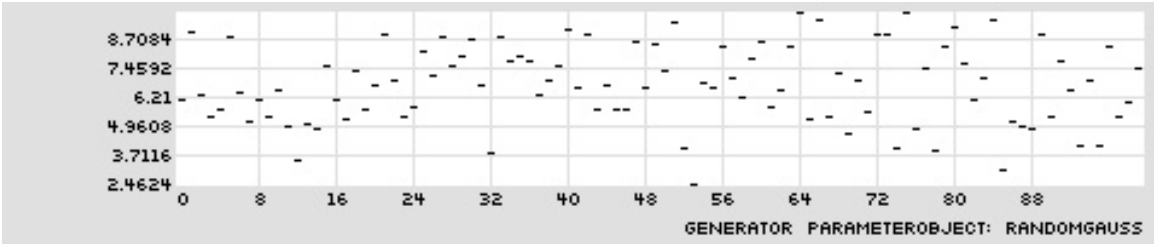
- RandomGauss: normal distribution, arguments mu and sigma
  - mu: center of distribution, between 0 and 1
  - sigma: deviation around center, where .001 is little deviation
  - mu at .3, sigma at .01, scaled between 0 and 10

```
pi{}ti{} :: tpmap 100 rg,.3,.01,0,10
randomGauss, 0.3, 0.01, (constant, 0), (constant, 10)
TPmap display complete.
```



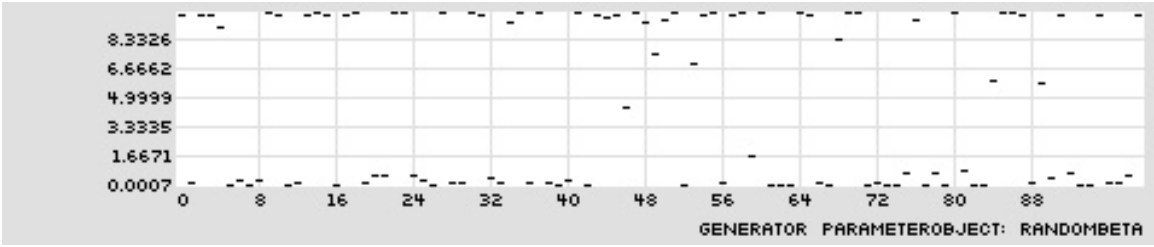
  - mu at .7, sigma at .2, scaled between 0 and 10

```
pi{}ti{} :: tpmap 100 rg,.7,.2,0,10
randomGauss, 0.7, 0.2, (constant, 0), (constant, 10)
TPmap display complete.
```



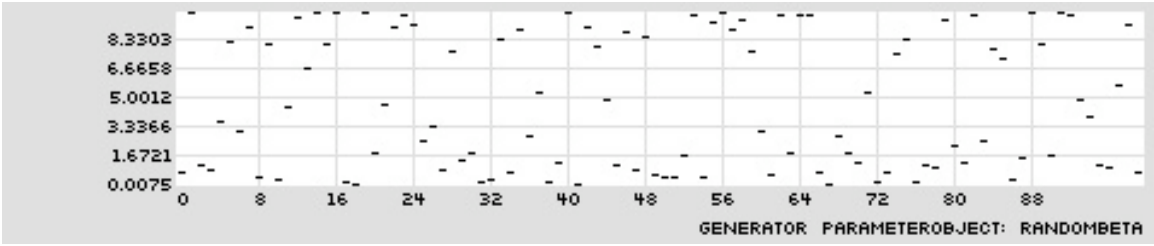
- RandomBeta: arguments alpha and beta
  - This implementation is different than Ames (1991)
  - alpha and beta: low values increase draw to boundaries
  - alpha and beta: large values approach a uniform distribution
  - alpha at .1, beta at .1, scaled between 0 and 10

```
pi{}ti{} :: tpmap 100 rb,0.1,0.1,0,10
randomBeta, 0.1, 0.1, (constant, 0), (constant, 10)
TPmap display complete.
```



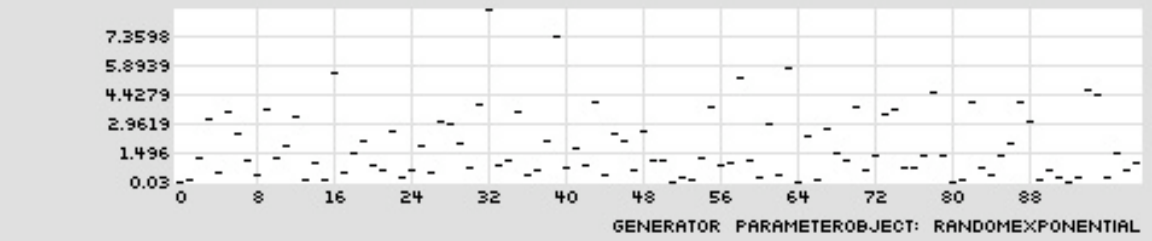
  - alpha at .3, beta at .3, scaled between 0 and 10

```
pi{}ti{} :: tpmap 100 rb,.3,.3,0,10
randomBeta, 0.3, 0.3, (constant, 0), (constant, 10)
TPmap display complete.
```



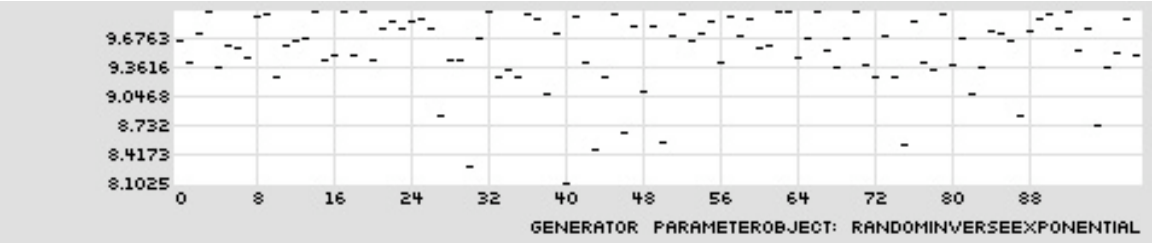
- RandomExponential and RandomInverseExponential
  - lambda: larger values create a tighter pull to to one boundary
  - exponential, lambda at 5, scaled between 0 and 10

```
pi{}ti{} :: tpmap 100 re,5,0,10
randomExponential, 5.0, (constant, 0), (constant, 10)
TPmap display complete.
```



- inverse exponential, lambda at 20, scaled between 0 and 10

```
pi{}ti{} :: tpmap 100 rie,20,0,10
randomInverseExponential, 20.0, (constant, 0), (constant, 10)
TPmap display complete.
```



- For all generators min and max can be embedded POs

3.10. Working with athenaCL

- Often best to use interactive mode for testing values, quick sketches, setting preferences
- Best to use a Python script for composing or other work
- Same preferences used in interactive mode are used in scripts
- For examples, the presence of the command prompt designates that athenaCL is in interactive mode

```
pi{}ti{} ::
```

3.11. Configuring Amplitudes

- Amplitudes in athenaCL are represented within the unit interval (0, 1)
- After creating texture, we can edit the amplitude with the TIe command
- The TIe command needs an argument for what Texture parameter to edit: enter “a” is for amplitude

- Parameter abbreviations can be found with the TIv command
- Setting the amplitude to a RandomUniform value between 0 and 1 [03a.py]

```
from athenaCL.libATH import athenaObj
ath = athenaObj.Interpreter()

ath.cmd('emo mp')

# create a new texture with instrument 45
ath.cmd('tin a 45')

# edit the amplitude of the texture to be RandomUniform between .1 and 1
ath.cmd('tie a ru,.1,1')
ath.cmd('eln')
ath.cmd('elh')
```

- Two parts, one with RandomUniform amplitudes, another with RandomExponential [03b.py]

  Note that textures have to have different names

```
from athenaCL.libATH import athenaObj
ath = athenaObj.Interpreter()

ath.cmd('emo mp')

# create a new texture with instrument 45
ath.cmd('tin a 45')
ath.cmd('tie a ru,.1,1')

# create a new texture with instrument 65
# texture must have a different name
ath.cmd('tin b 65')
ath.cmd('tie a re,15,.2,1')

ath.cmd('eln')
ath.cmd('elh')
```

- Three parts, RandomUniform, RandomExponential, and RandomBeta amplitudes [03c.py]

```
from athenaCL.libATH import athenaObj
ath = athenaObj.Interpreter()

ath.cmd('emo mp')

# create a new texture with instrument 45
ath.cmd('tin a 45')
ath.cmd('tie a ru,.1,1')

# create a new texture with instrument 65
ath.cmd('tin b 65')
ath.cmd('tie a re,15,.2,1')

# create a new texture with instrument 53
ath.cmd('tin c 53')
ath.cmd('tie a rb,.1,.1,.3,.7')

ath.cmd('eln')
ath.cmd('elh')
```

3.12. Duration and Sustain

- Duration
  - The temporal space of an event
  - If events are packed end to end, the time of the next event
  - If a notated event, the written rhythm
- Sustain
  - The sounding (actual) time of the event
  - A scalar applied to the duration
  - A scalar of 0.2 would suggest a staccato (shortened) event
  - A scalar of 1.2 would create overlapping events

3.13. The Pulse Triple

- athenaCL supports both absolute and relative rhythm values
- The PulseTriple is relative to the beat-defining tempo and made of three values
  - Divisor: divides the tempo beat duration
  - Multiplier: scales the value divided
  - Accent: a rhythm-specific amplitude value, between 0 (o) and 1 (+) (or with symbolic dynamics: mp, mf, etc)
- Conventional rhythms can be easily expressed
  - (4,1,1): 1/4th of a beat (if the beat is a quarter, a sixteenth note)
  - (4,3,1): 3/4ths of a beat (if the beat is a quarter, a dotted eighth note)
  - (1,4,1): 4 beats (if the beat is a quarter, a whole note)
  - (3,1,1): 1/3rd of a beat (if the beat is a quarter, a triplet eighth)
  - (5,8,1): 8/5ths of a beat
- Representational redundancy may be useful
  - (4,2,1) is the same as (2,1,1)

- (1,5,1) is the same as (4,20,1)

3.14. Basic Rhythm ParameterObjects

- PulseTriple: create PulseTriples from embedded ParameterObjects

```
pi{}ti{} :: tpv pulsetriple
Rhythm Generator ParameterObject
{name,documentation}
PulseTriple         pulseTriple, parameterObject, parameterObject, parameterObject,
parameterObject
                    Description: Produces Pulse sequences with four Generator
ParameterObjects that
                    directly specify Pulse triple values and a sustain scalar. The
Generators specify
                    Pulse divisor, multiplier, accent, and sustain scalar. Floating-
point divisor and
                    multiplier values are treated as probabilistic weightings. Note:
divisor and
                    multiplier values of 0 are not permitted and are replaced by 1;
the absolute value
                    is taken of all values. Arguments: (1) name, (2) parameterObject
{pulse divisor},
                    (3) parameterObject {pulse multiplier}, (4) parameterObject
{accent value between 0
                    and 1}, (5) parameterObject {sustain scalar greater than 0}
```

- ConvertSecond: create durations form values in seconds

```
pi{}ti{} :: tpv cs
Rhythm Generator ParameterObject
{name,documentation}
ConvertSecond       convertSecond, parameterObject
                    Description: Allows the use of a Generator ParameterObject to
create rhythm
                    durations. Values from this ParameterObject are interpreted as
equal Pulse duration
                    and sustain values in seconds. Accent values are fixed at 1.
Note: when using this
                    Rhythm Generator, tempo information (bpm) has no effect on event
timing. Arguments:
                    (1) name, (2) parameterObject {duration values in seconds}
```

3.15. Configuring Rhythms

- After creating texture, we can edit the rhythm with the TIe command
- The TIe command needs an argument for what Texture parameter to edit: enter “r” for rhythm
- Using basketGen to control the multiplier [03d.py]

```
from athenaCL.libATH import athenaObj
ath = athenaObj.Interpreter()

ath.cmd(“emo mp”)

ath.cmd(“tin a 45”)
```

```
ath.cmd("tie a rb,.3,.3,.5,.8")
ath.cmd("tie r pt,(c,4),(bg,oc,(3,3,2)),(c,1)")

ath.cmd("eln")
ath.cmd("elh")
```

- Using two basketGens to control multiplier and divisor independently [03e.py]

```
from athenaCL.libATH import athenaObj
ath = athenaObj.Interpreter()

ath.cmd("emo mp")

ath.cmd("tin a 45")
ath.cmd("tie a rb,.3,.3,.4,.8")
ath.cmd("tie r pt,(c,4),(bg,oc,(3,3,2)),(c,1)")

ath.cmd("tin b 65")
ath.cmd("tie a re,15,.3,1")
ath.cmd("tie r pt,(bg,rp,(2,1,1,1)),(c,1),(c,1)")

ath.cmd("eln")
ath.cmd("elh")
```

- Using two basketGens to control multiplier and divisor independently [03f.py]

```
from athenaCL.libATH import athenaObj
ath = athenaObj.Interpreter()

ath.cmd("emo mp")

ath.cmd("tin a 45")
ath.cmd("tie a rb,.3,.3,.4,.8")
ath.cmd("tie r pt,(c,4),(bg,oc,(3,3,2)),(c,1)")

ath.cmd("tin b 65")
ath.cmd("tie a re,15,.3,1")
ath.cmd("tie r pt,(bg,rp,(2,1,1,1)),(c,1),(c,1)")

ath.cmd("tin c 67")
ath.cmd("tie a rb,.1,.1,.4,.6")
ath.cmd("tie r cs,(rb,.2,.2,.01,1.5)")

ath.cmd("eln")
ath.cmd("elh")
```

3.16. Configuring Time Range

- After creating texture, we can edit the time range with the TIe command
- The TIe command needs an argument for what Texture parameter to edit: enter "t" for time range
- Enter two values in seconds separated by a comma
- Staggering the entrances of three parts [03g.py]

```
from athenaCL.libATH import athenaObj
```

```
ath = athenaObj.Interpreter()

ath.cmd("emo mp")

ath.cmd("tin a 45")
ath.cmd("tie t 0,20")
ath.cmd("tie a rb,.3,.3,.4,.8")
ath.cmd("tie r pt,(c,4),(bg,oc,(3,3,2)),(c,1)")

ath.cmd("tin b 65")
ath.cmd("tie t 10,20")
ath.cmd("tie a re,15,.3,1")
ath.cmd("tie r pt,(bg,rp,(2,1,1,1)),(c,1),(c,1)")

ath.cmd("tin c 67")
ath.cmd("tie t 15,25")
ath.cmd("tie a rb,.1,.1,.4,.6")
ath.cmd("tie r cs,(rb,.2,.2,.01,1.5)")

ath.cmd("eln")
ath.cmd("elh")
```

3.17. Musical Design Report 1

- Must be primarily rhythmic in nature
- Must employ at least 4 different timbre sources
- Should have at least an AB or ABA form
- Must prominently feature both the beta and exponential distributions
- Can be composed with athenaCL, athenaCL and other tools, or other tools alone
- See syllabus for details on other aspects

3.18. Digital Audio Workstations

- The merger of software for editing MIDI and notation with software for editing digital audio
- Numerous commercial varieties: ProTools, Digital Performer, Cubase, FL, Logic, GarageBand
- Inexpensive varieties: Reaper
- Free varieties: Ardour, Rosegarden
- Having access to a DAW with virtual instruments will greatly assist your projects in this class

3.19. Digital Audio Workstations: Importing and Mixing Digital Audio

- Create tracks to store audio

- Drag and drop digital audio into a track
- Adjust levels, process, and edit
- Bounce to disc to mix down to a single audio file

3.20. Digital Audio Workstations: Importing MIDI and Rendering Digital Audio

- Create tracks to store MIDI or for virtual instruments
- Drag and drop MIDI into a track
- Render, freeze, or bounce realization of virtual instrument

Chapter 4. Meeting 4, Foundations: Historical and Categorical Perspectives

4.1. Announcements

- Musical Design Report 1 due Tuesday, 23 February

4.2. Reading: Ames: Automated Composition in Retrospect: 1956-1986

- Ames, C. 1987. "Automated Composition in Retrospect: 1956-1986." *Leonardo* 20(2): 169-185.
- Is it surprising that Ames writes: "it is therefore not surprising that these developments have met with continuing -- and often virulent -- resistance" (1987, p. 169)?
- How was the DATATRON used to generate a melody?
- How was MUSICOMP different from the work on the Illiac Suite?
- How does Ames isolate the contribution of Koenig and Xenkakis as contributing to modularity in system design?
- What trends does Ames describe in systems that were contemporary to his article?

4.3. Reading: Ariza: Navigating the Landscape of Computer-Aided Algorithmic Composition Systems: A Definition, Seven Descriptors, and a Lexicon of Systems and Research

- Ariza, C. 2005b. "Navigating the Landscape of Computer-Aided Algorithmic Composition Systems: A Definition, Seven Descriptors, and a Lexicon of Systems and Research." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 765-772. Internet: http://www.flexatone.net/docs/nlcaacs.pdf.
- What is the definition of CAAC proposed in this article?
- Why does the definition of CAAC exclude notation software and DAWs?
- What are the seven descriptors proposed, and which seem the most important?

Chapter 5. Meeting 5, History: Serialism, Loops, Tiling, and Phasing

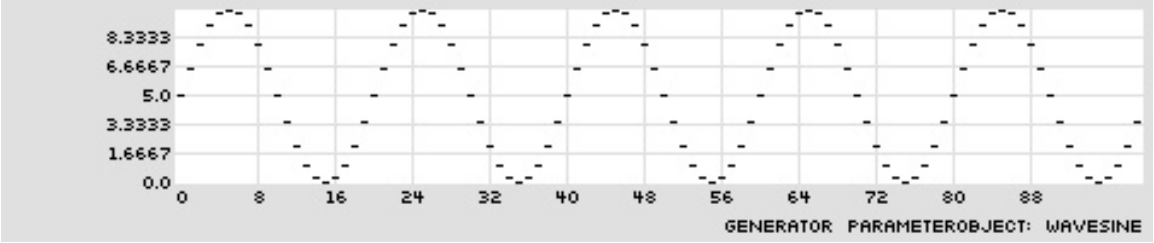
5.1. Announcements

- Musical Design Report 1 due Tuesday, 23 February
- Review readings from last class

5.2. Trigonometric Functions and Break-Point Graphs as ParameterObjects

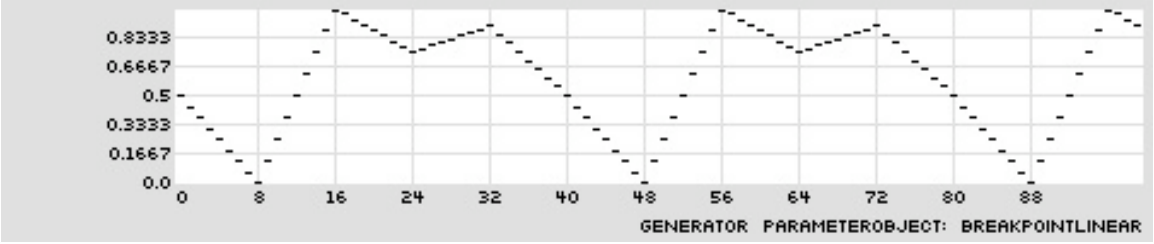
- WaveSine: A scalable sine oscillator controlled by seconds or events per cycle

```
pi{}ti{} :: tpmap 100 ws,e,20,0,0,10
waveSine, event, (constant, 20), 0, (constant, 0), (constant, 10)
TPmap display complete.
```



- BreakPointLinear: Break point segments defined by seconds or events

```
pi{}ti{} :: tpmap 100 bpl,e,l,((0,.5),(8,0),(16,1),(24,.75),(32,.9),(40,.5))
breakPointLinear, event, loop, ((0,0.5),(8,0),(16,1),(24,0.75),(32,0.9),(40,0.5))
TPmap display complete.
```



- Numerous alternative trigonometric function generators exist as ParameterObjects: WaveCosine, WavePulse, WaveSawDown, WaveSine, WaveTriangle
- Numerous alternative break-point function generators exist as ParameterObjects: BreakPointFlat, BreakPointHalfCosine, BreakPointLinear, BreakPointPower

5.3. Configuring Tempo

- The TIe command can be use to edit tempo by specifying "b" for BPM
- Tempo can be controlled by any ParameterObject

5.4. Approaches to Composing Time

- Creating overlapping repeats of the same material
- Creating overlapping repeats of transformed material
- Creating ordered material that is then transformed in ways that retain order

5.5. Canons and Tiling

- Create an initial line and repeat it with staggered entrances
- An approach to polyphony
- The initial line can be temporally shifted and temporally transformed
- Can be seen as an approach to musical tiling

5.6. Listening: Andriessen

- Louis Andriessen (1939-)
- Dutch composer notable for combining American Minimalism with (at times) more diverse harmonic language
- Andriessen: "Hout" (1991)

5.7. Building a Basic Beat

- Kick, snare, and hats
- Command sequence:
  - emo mp
  - tin a 36
  - tie r pt,(c,2),(bg,oc,(7,5,2,1,1)),(c,1)
  - tin b 37
  - tie r pt,(c,2),(bg,oc,(3,5)),(bg,oc,(0,1))
  - tin c 42
  - tie r pt,(c,2),(c,1),(bg,oc,(0,1))
  - eln; elh

5.8. A Basic Beat with More Complex Snare Part

- Continued command sequence:
  - tio b
  - tie r pt,(c,4),(bg,rp,(3,3,5,4,1)),(bg,oc,(0,1,1))
  - eln; elh

5.9. Adding Canonic Snare Imitation: Texture Copying

- Copying a texture creates a new, independent, and dynamic part
- While having identically configured ParameterObjects, if randomness is employed, unique structures will be created
- Continued command sequence:
  - tio b
  - ticp b b1
  - tie t .25, 20.25

- tie i 76
- ticp b b2
- tie t .5, 20.5
- tie i 77
- eln; elh

5.10. Saving and Loading the AthenaObject

- An athenaCL XML file can be loaded in to athenaCL to restore Textures
- These XML files can be automatically created whenever an event list is created
- Continued command sequence:
  - eoo xao
  - eln

5.11. Building an Extended Rhythmic Line with Canonic Imitation

- Using different length ordered cyclic generators will create complex but non-random sequences
- Command sequence:
  - aorm confirm
  - emo mp
  - tin a 77
  - tie r pt,(c,1),(c,1),(c,1)
  - tin b 67
  - tie r pt,(bg,oc,(2,4,1)),(bg,oc,(3,5,1,7,1,3)),(c,1)
  - ticp b b1
  - tie t 0.125,20.125
  - tie i 60
  - ticp b b2

- tie t 0.25,20.25
- tie i 68
- eln; elh

5.12. Creating Mensural Canons

- Mensural canons use ratio-base time signatures for each part
- Continued command sequence:
  - tio b1
  - tie b c,90
  - tio b2
  - tie b c,180
  - eln; elh

5.13. Extensions

- We can generate complex, deterministic patterns by combining cycles at high ratios
- The same musical rhythm at different (low ratio related) rates produces interesting musical results

5.14. Tonal, Atonal, and Post-Tonal

- Tonal music employs functional harmony
  - Harmonies (chords) have a trajectory, expectation, and a resolution
  - One (or two) chords are more than others
- Atonal music does not employ functional harmony
  - The expectations and priorities of chords are removed
  - Ideally, no pitch is more important than any other
- Post-tonal refers approaches to harmony other than tonal
  - May be atonal, or may employ other approaches to pitch
  - Pitch centers may be developed and exploited

5.15. Serialism

- An approach to atonality that serialized (ordered) elements of musical parameters, developed by Arnold Schoenberg
- An alternative approach to atonality employed chords that completed the aggregate (all 12 pitches), developed by Josef Matthias Haur
- By serializing the order of all 12-tone pitches, all get equal usage
- Pitch groups smaller than 12 can be used
- A series of all 12 tones is used as a motivic origin
  - The series can be transposed to any of 12 pitch levels: prime
  - The series can be reversed: retrograde
  - The series can be inverted ((12-*n*) % 12): inversion
  - The inverted series can be reversed: retrograde inversion
  - The 12 x 4 possible rows can be presented in a matrix

    Generated with Python tools in music21: http://code.google.com/p/music21/

```
from music21 import serial

p = [8,1,7,9,0,2,3,5,4,11,6,10]
print serial.rowToMatrix(p)
```

```
  0  5 11  1  4  6  7  9  8  3 10  2
  7  0  6  8 11  1  2  4  3 10  5  9
  1  6  0  2  5  7  8 10  9  4 11  3
 11  4 10  0  3  5  6  8  7  2  9  1
  8  1  7  9  0  2  3  5  4 11  6 10
  6 11  5  7 10  0  1  3  2  9  4  8
  5 10  4  6  9 11  0  2  1  8  3  7
  3  8  2  4  7  9 10  0 11  6  1  5
  4  9  3  5  8 10 11  1  0  7  2  6
  9  2  8 10  1  3  4  6  5  0  7 11
  2  7  1  3  6  8  9 11 10  5  0  4
 10  3  9 11  2  4  5  7  6  1  8  0
```

- Milton Babbitt and Pierre Boulez extended serial techniques to new parameters and alternative organizations
- Karlheinz Stockhausen and others attempted to employ serial techniques to organize parameters in the early Electronic Music studio
- Total serialism orders amplitudes, rhythms, and other musical parameters

5.16. Listening: Boulez

- Pierre Boulez (1925-)
- Post WWII and total serialism
- Boulez: "Structures, Book I" (1952)

5.17. Extensions

- The algorithmic opportunities of serialism led many composers to generalize such techniques with the computer
- athenaCL features Paths as a way for Textures to share source Pitch data
- One Path might be shared by multiple Textures, each transposing, reversing, and inverting this Path to create serial arrangements
- While some have tried (Babbitt 1958), serial rhythm techniques have not been widely embraced

5.18. Phasing

- Musical material shifting in and out of time, or moving at different rates
- Developed out of manipulations to recording reels: flanging and phasing

- Can be used as a canon-like technique

5.19. Listening: Reich

- Steve Reich (1936-)
- Influenced by techniques of minimalism based in part on music of Terry Riley, La Monte Young, and others
- Reich: "It's gonna rain" (1965)
- "Scorification" of a technological process for acoustic instruments
- Reich: "Piano Phase" (1967)

5.20. Phasing with athenaCL Python Libraries

- pianoPhase.py

```
import os
from athenaCL.libATH import midiTools
from athenaCL.libATH import osTools
from athenaCL.libATH import pitchTools
from athenaCL.libATH import rhythm
from athenaCL.libATH.libOrc import generalMidi
from athenaCL.libATH.libPmtr import parameter

OUTDIR = '/Volumes/xdisc/_scratch'
BEATDUR = rhythm.bpmToBeatTime(225) # provide bpm value

def getInstName(nameMatch):
   for name, pgm in generalMidi.gmProgramNames.items():
      if name.lower().startswith(nameMatch.lower()):
         return pgm # an integer
   return None

def getSource(repeat):
   """get source melody and rhythm"""

   pitchSequence = ['E4','F#4','B4','C#5','D5','F#4',
                    'E4','C#5','B4','F#4','D5','C#5']
   rhythmSequence = [.5, .5, .5, .5, .5]
   ampGen = parameter.factory(['ws','e',14,0,90,120]) # sine osc b/n 90 and 120
```

```
    score = []
    tStart = 0.0
    for i in range(len(pitchSequence) * repeat):
       ps = pitchTools.psNameToPs(pitchSequence[i%len(pitchSequence)])
       pitch = pitchTools.psToMidi(ps)
       dur = BEATDUR * rhythmSequence[i%len(rhythmSequence)]
       amp = int(round(ampGen(0)))
       pan = 30
       event = [tStart, dur, amp, pitch, pan]
       score.append(event)
       tStart = tStart + dur

    return score, len(pitchSequence)

 def transformSource(score, srcLength):
    """transform source, srcLength is size of each melodic unit
    """
    post = []
    octaveShift = -1
    panShift = 60
    shiftUnit = BEATDUR / 16.

    eCount = 0
    repCount = 0 # starting at zero means first cycle will be in phase
    for event in score:
       if eCount % srcLength == 0:
          shift = shiftUnit * repCount
          repCount = repCount + 1 # increment after using

       newEvent = [event[0]+shift, event[1], event[2],
                   event[3]+(octaveShift*12), (event[4]+panShift)%128]
       post.append(newEvent)
       eCount = eCount + 1 # increment for each event
    return post

 def main():
    repeat = 33
    partA, seqLen = getSource(repeat)
    partB = transformSource(partA, seqLen)

    trackList = [('part-a', getInstName('piano'), None, partA),
                 ('part-b', getInstName('piano'), None, partB),]
    path = os.path.join(OUTDIR, 'test.midi')
    mObj = midiTools.MidiScore(trackList)
    mObj.write(path)
    osTools.openMedia(path)

 if __name__ == '__main__':
    main()
```

5.21. Beats with athenaCL Python Libraries

- basicBeat.py

```
import os, random
from athenaCL.libATH import midiTools
from athenaCL.libATH import osTools
from athenaCL.libATH import pitchTools
from athenaCL.libATH import rhythm
from athenaCL.libATH.libOrc import generalMidi
from athenaCL.libATH.libPmtr import parameter
```

```
OUTDIR = '/Volumes/xdisc/_scratch' # provide output directory
BEATDUR = rhythm.bpmToBeatTime(160) # provide bpm value

def getInstPitch(nameMatch):
   for name, pgm in generalMidi.gmPercussionNames.items():
      if name.lower().startswith(nameMatch.lower()):
         return pgm # an integer
   raise NameError('bad pitch name')

def getKickSnare(repeat):
   rhythmA = [1, 1.5, .5, 1]
   rhythmB = [1.5, .5, 1.5, .5]
   rhythmC = [1.75, .25,  1.5, .125, .125, .125, .125]
   instA   = ['acousticBassDrum','sideStick']
   instB   = ['sideStick']

   ampGen = parameter.factory(['rb',.2,.2,110,127])

   score = []
   tStart = 0.0
   for q in range(repeat):
      if q % 3 == 0:
         rhythmSequence = rhythmB
         instSequence = instA
      elif q % 11 == 10:
         rhythmSequence = rhythmC
         instSequence = instB
         random.shuffle(rhythmSequence)
      else:
         rhythmSequence = rhythmA
         instSequence = instA

      for i in range(len(rhythmSequence)):
         inst = instSequence[i % len(instSequence)]
         pitch = getInstPitch(inst)
         dur = BEATDUR * rhythmSequence[i % len(rhythmSequence)]
         amp = int(round(ampGen(0)))
         pan = 63
         event = [tStart, dur, amp, pitch, pan]
         score.append(event)
         tStart = tStart + dur
   return score, len(rhythmSequence)

def getHats(repeat):
   rhythmSequence = [.5, .5, .25, .25, .5, .5, .5, .5]
   instSequence   = ['closedHiHat','closedHiHat',
                     'closedHiHat','closedHiHat',
                     'closedHiHat','openHiHat']
   ampGen = parameter.factory(['rb',.2,.2,50,80])

   score = []
   tStart = 0.0
   for q in range(repeat):
      for i in range(len(rhythmSequence)):
         inst = instSequence[i % len(instSequence)]
         pitch = getInstPitch(inst)
         dur = BEATDUR * rhythmSequence[i % len(rhythmSequence)]
         amp = int(round(ampGen(0)))
         pan = 63
         event = [tStart, dur, amp, pitch, pan]
         score.append(event)
         tStart = tStart + dur

   return score, len(rhythmSequence)
```

```
def main():
   repeat = 33
   partA, seqLen = getKickSnare(repeat)
   partB, seqLen = getHats(repeat)

   trackList = [('part-a', 0, 10, partA),
                ('part-b', 0, 10, partB),]

   path = os.path.join(OUTDIR, 'test.midi')
   mObj = midiTools.MidiScore(trackList)
   mObj.write(path) # writes in cwd
   osTools.openMedia(path)

if __name__ == '__main__':
   main()
```

5.22. Building an Extended Rhythmic Line with Fixed Tempo Phasing

- Using different tempi will create shifting rhythmic patterns
- Command sequence:
  - aorm confirm
  - emo mp
  - tin a 70
  - tie r pt,(bg,oc,(2,4,4)),(bg,oc,(4,1,1,2,1)),(c,1)
  - tie t 0,60
  - ticp a a1
  - tie b c,124
  - ticp a a2
  - tie b c,128
  - eln; elh

5.23. Building an Extended Rhythmic Line with Dynamic Tempo Phasing

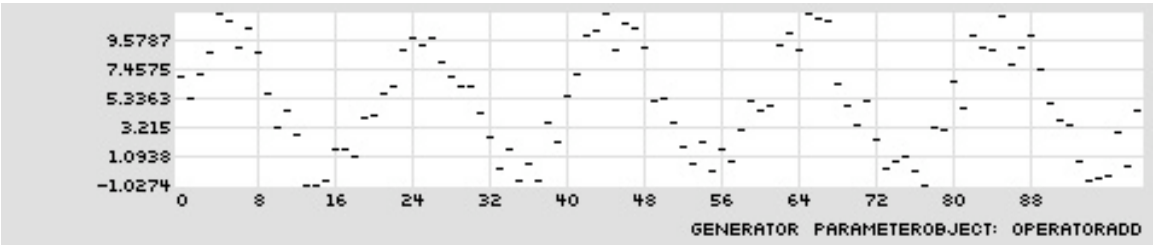
- Oscillating the tempo at different rates will create dynamic changes
- Command sequence:
  - aorm confirm

- emo mp
- tin a 64
- tie r pt,(bg,oc,(2,4,4)),(bg,oc,(4,1,1,2,1)),(c,1)
- tie t 0,60
- ticp a a1
- tie i 60
- tie b ws,t,20,0,115,125
- ticp a a2
- tie i 69
- tie b ws,t,30,0,100,140
- eln; elh

5.24. Extensions

- Many works have been built with slow and gradual tempo changes
- Tempos might slowly deviate with a BreakPointLinear or similar generator
- Tempos might be randomly perturbed by adding in randomness: PO OperatorAdd can sum two ParameterObjects

```
pi{}ti{} :: tpmap 100 oa,(ws,e,20,0,0,10),(ru,-2,2)
operatorAdd, (waveSine, event, (constant, 20), 0, (constant, 0), (constant, 10)),
(randomUniform,
(constant, -2), (constant, 2))
TPmap display complete.
```



Chapter 6. Meeting 6, Workshop

6.1. Announcements

- Musical Design Report 1 due Today, 23 February
- Quiz on Thursday
- Download Martingale:

  http://code.google.com/p/martingale/

6.2. Workshop: Musical Design Report 1

- `Four students will present their reports today`

6.3. Installing and Configuring Csound

- Download and install most recent Csound 5

  http://sourceforge.net/projects/csound/files/

- Test installation
  - Windows: run Csound.exe
  - Others: open a terminal, enter: csound

6.4. Testing Csound in athenaCL

- athenaCL can write separate score and orchestra files, or a combined .scd file; depends on EventOutput settings (select csd with EOo)
- athenaCL may need to have a user preference set for where the Csound binary is located (use the APea command)
- athenaCL will create a batch file (.bat) to automate rendering of Csound files to audio
- The audio file, after rendering, will be stored and named in the same location as other output files
- Command sequence:

    - emo cn
    - tin a 82
    - tie x6 ws,e,14,0,200,16000
    - eln
    - elr
    - elh
- With the ELauto command, rendering (ELr) and hearing (ELh) can be automatically executed following the use of ELn

6.5. Testing PD and Martingale

- Download and install PD-Extended

  http://puredata.info/downloads

- Download Martingale manually:

  http://code.google.com/p/martingale/

- Place martingale anywhere on your file system
- Add the "martingale/pd/lib" directory to Preferences > Path; this permits loading abstractions from the martingale library
- Open pd/demo/earLimits.pd
- Make sure "compute audio" is on, click check boxes, and select frequencies

Chapter 7. Meeting 7, History: Gottfried Michael Koenig

7.1. Announcements

- Test direct rendering of CSD files with Csound if ELr is not working
- Make sure you have PD-extended installed and Martingale on your system

7.2. Quiz

- 10 Minutes

7.3. Gottfried Michael Koenig

- Gottfried Michael Koenig (1926-)
- 1954-1964: Worked with Stockhausen and others at West German Radio in Cologne
- Composed for tape and acoustic instruments
- 1963-1964: Studied programming, began developing software for CAAC
- 1964-1986: Director of the Institute of Sonology in the Netherlands
- Employed CAAC at three different levels
  - Symbolic: output from computer used to transcribe notation
  - Control: create sequences of control voltage mapped to synthesis parameters
  - Direct: employed direct creation of waveforms to create non-standard synthesis techniques

7.4. Reading: Koenig: The Use of Computer Programs in Creating Music

- Koenig, G. M. 1971. "The Use of Computer Programs in Creating Music." In *Music and Technology (Proceedings of the Stockholm Meeting organized by UNESCO)*. Paris: La Revue Musicale. 93-115. Internet: http://www.koenigproject.nl/Computer_in_Creating_Music.pdf.
- Koenig states that the use of the computer does not herald a new musical epoch: instead, what does he see the computer as offering?
- Koenig describes a variable function generator: what is this?

- Koenig sees work in the electronic music studio as suggesting some of the practices of algorithmic composition: how so?
- Koenig introduces the term composition theory: what might this mean?
- What role did Koenig imagine for the computer in the work of composers and music students?
- Koenig describes a technique of "sound production": what is this?

7.5. Koenig: CAAC for Acoustic Instruments

- Two early software systems
- 1964: Project 1 (PR1)
- 1969: Project 2 (PR2)
- Favored discrete value generation and selection

7.6. PR1: Concepts

- 1964: Project 1 (PR1): Programmed in Fortran for the IBM 7090
- A closed system, providing output based on user parameters
- A user specified six tempo values, twenty-eight entry delays (rhythmic values), a random generator seed value, and the length of the composition
- Materials were algorithmically selected
    - Series: random permutations, selection without replacement
    - Alea: random selection
- Koenig saw series generation as an abstraction of twelve-tone techniques: "the need for variation is satisfied without there having to be the pretense that somewhere deep inside the work the twenty-fifth permutation is still being systematically derived from an original series" (1970a, p. 34).
- At a larger level, 12-tone rows are created and deployed and three-note aggregate completing trichords are created.
- A seven-section formal structure controls the large-scale form, defining a position in a range from regular/periodic to irregular/aperiodic.
- Output is provided for six parameters: (1) timbre (instrument or instrument group), (2) rhythm, (3) pitch, (4) sequence, (5) octave register, and (6) dynamic.

- Sequence is spare parameter, applied to chord formation or timbre component within a timbre group
- All parameters are independent

7.7. PR2: Concepts

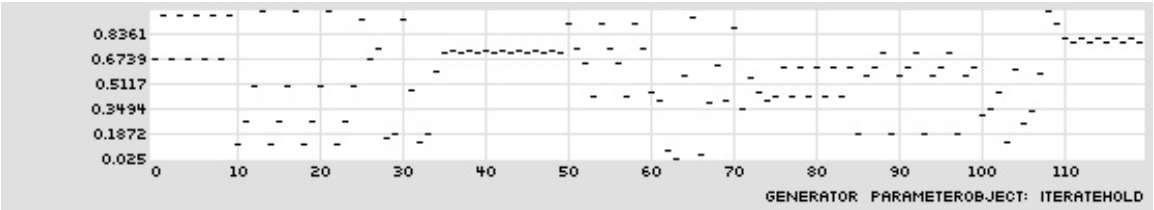
- 1969: Project 2 (PR2): Algol, then Fortran for the PDP-15
- A closed system, but more general and user-configurable
- Eight parameters are generated: (1) instrument, (2) harmony, (3) register, (4) entry delay, (5) duration, (6) rest, (7) dynamics, and (8) mode of performance.
- Expanded tools for algorithmic selection
  - Series
  - Alea
  - Ratio: weighted random selection
  - Group: repetition of values
  - Sequence: ordered selection
  - Tendency: random selection within dynamic boundaries

7.8. PR2: The List-Table-Ensemble Principle

- Selection procedures can be used on user-specified numeric or symbolic values (lists, stockpiles, or tables), or new, algorithmically generated expansions of user-specified numeric or symbolic values (ensembles).
- Lists: raw stockpiles of data (assigned index values for access)
- Tables: user-organized collection of indexes pointing to data in Lists
- Ensembles: selection methods are used to create intermediary groups of data that are then drawn from to produce parameter values
- A techniques of meta-selection that constrains values within distinct representations (distributions and orderings)
- IterateHold: a rough analogy to the list-table-ensemble principle: select a number values from a PO, employ these for a number of times, and then regenerate a new selection

```
:: tpmap 120 ih,(ru,0,1),(bg,oc,(2,4,13)),(bg,oc,(10,15))
```

```
iterateHold, (randomUniform, (constant, 0), (constant, 1)), (basketGen,
orderedCyclic, (2,4,13)), (basketGen, orderedCyclic, (10,15)), orderedCyclic
TPmap display complete.
```



7.9. Listening: Koenig

- Three Asko Pieces
- Koenig: *Three Asko Pieces* (1982)

7.10. PR2 Selection Principles: Ratio

- Weighted randomness can be achieved by configuration of BasketGen values
- More control can obtained by configuring a zero-order Markov chain, to be discussed later

7.11. Controlling Pitch in athenaCL

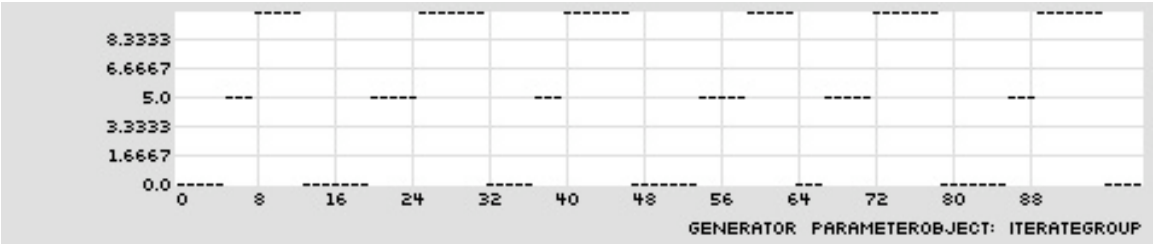
- Paths provide ordered collections of pitch groups (Multisets) with proportional durations

- A Texture is assigned a Path based on the last-created Path, an assigned Path, or an automatically created Path (if none exist)
- The default Path is a single pitch, C4
- A Texture can transform the Path with ParameterObjects assigned to the field (transposition) and octave (register shift) parameters
- Different TextureModules can deploy Paths in very diverse ways

7.12. PR2 Selection Principles: Group

- IterateGroup: Two POs, one generating values, the other selecting how many times those values are repeated before a new selection is made

```
:: tpmap 100 ig,(bg,oc,(0,5,10)),(bg,rc,(3,5,7))
iterateGroup, (basketGen, orderedCyclic, (0,5,10)), (basketGen, randomChoice,
(3,5,7))
TPmap display complete.
```



- Create a collection of values, select a value, and then repeat a selected number of times
- Command sequence:
  - emo m
  - tin a 6
  - tie r cs,(rb,.2,.2,.02,.25)
  - tie f ig,(bg,rc,(2,4,7,9,11)),(bg,rp,(2,3,5,8,13))
  - tie o ig,(bg,oc,(-2,-1,0,1)),(ru,20,30)
  - ticp a b c d
  - eln; elh

7.13. PR2 Selection Principles: Tendency Mask

- Random values selected from within dynamic minimum and maximum value range
- Can be implemented with any Generator PO that has min/max parameter
- Boundaries can be controlled by BreakPoint, Wave, or similar ParameterObjects
- A powerful technique for creating long range behavior
- Here, a break-point function and a wave sine generator form the boundaries of a random beta selection to control pitch
- Command sequence:
  - emo m
  - tin a 15
  - tie r cs,(ig,(ru,.01,.25),(ru,4,12))
  - tie a ru,.2,(cg,u,.3,.9,.005)
  - tie f rb,.2,.2,(bpl,t,l,((0,-12),(30,12))),(ws,t,29,0,0,24)
  - eln; elh
- A powerful technique for creating long range behavior
- Here, random octave values are chosen between two wave triangle generators
- Command sequence:
  - emo m
  - pin a d,e,g,a,b
  - tin a 107
  - tie r pt,(c,16),(ig,(bg,rc,(1,2,3,5,7)),(bg,rc,(3,6,9,12))),(c,1)
  - tie o ru,(wt,t,25,0,-2,4),(wt,t,20,0,-3,1)
  - eln; elh

7.14. Reading: Koenig: Aesthetic Integration of Computer-Composed Scores

- Koenig, G. M. 1983. "Aesthetic Integration of Computer-Composed Scores." *Computer Music Journal* 7(4): 27-32.

- Koenig states that "... to react functionally means ... to refrain from imitation of a particular production mode in another medium": what is he suggesting?

- What is Koenig suggesting about the use of histograms, where the composer supplies histograms and the computer program takes care of the data connections?

- What is the process of transcription that Koenig describes? How is this different than conventional transcription?

- What is aesthetic integration? Does Koenig suggest that this step can also be automated?

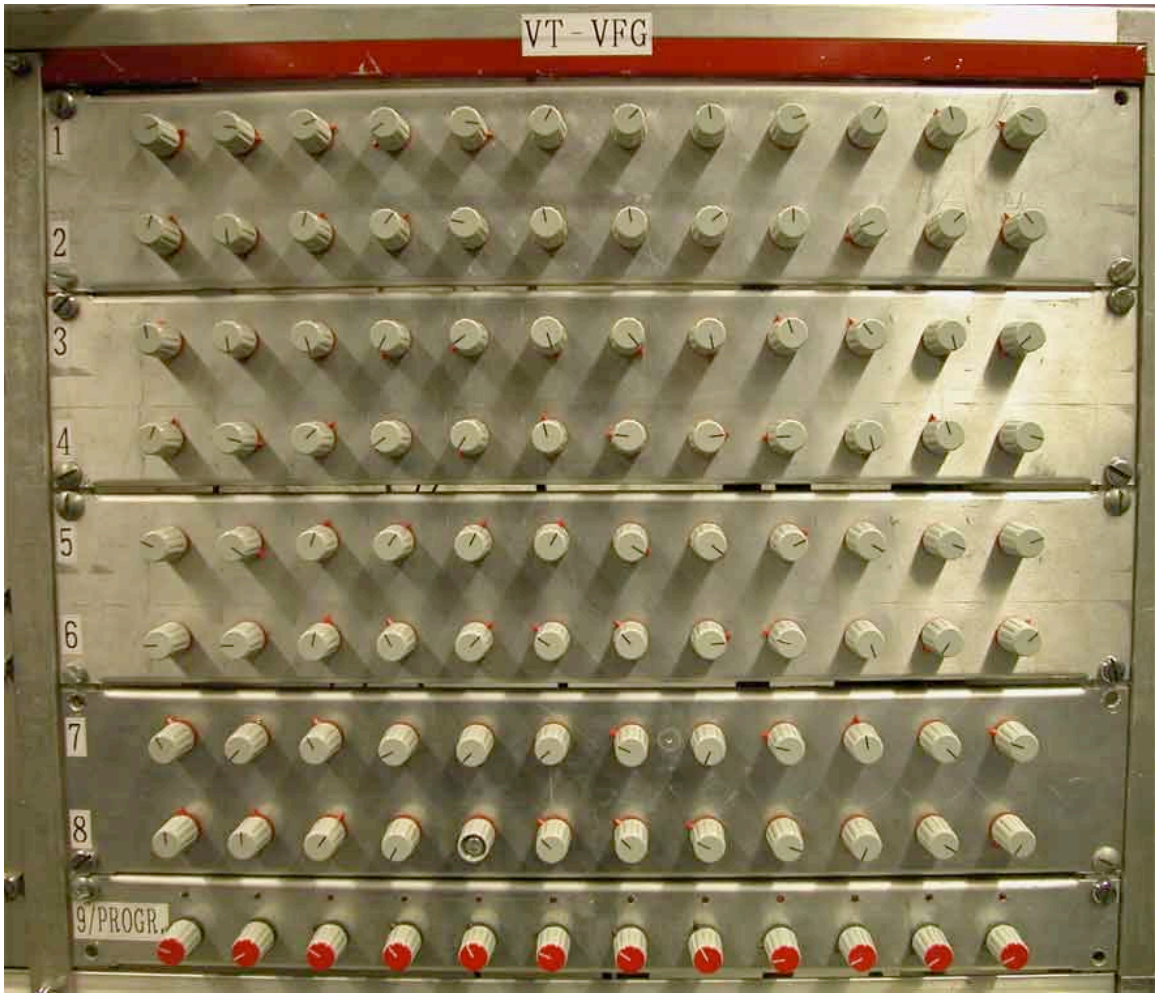
- Koenig talks about composer having a sense of responsibility for the aesthetic result: why is this significant?

7.15. Koenig: CAAC for Voltage Control

- Used PR1 to generate events that were encoded in voltage control data

- Voltage control data processed and translated to various musical parameters at different speeds

- Used "variable function generator" (1966) to set and deploy values from the control rate to the audio rate


VT - VFG
1
2
3
4
5
6
7
8
9/PROGR.

- Produced Funktion pieces with this method: Funktion Grün (1967), Funktion Gelb (1968), Funktion Orange (1968), Funktion Rot (1968), Funktion Blau (1969), Funktion Indigo (1969), Funktion Violett (1969), Funktion Grau (1969)

- Similar methods will be employed by outputting athenaCL generators to PD Arrays

7.16. Listening: Koenig

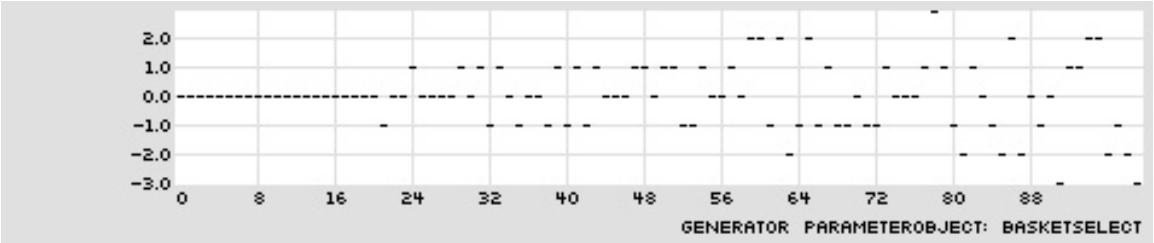
- Employed techniques of Funktion pieces

- Koenig: "Terminus X" (1967)

7.17. Alternative Approaches to Grouping and Masking

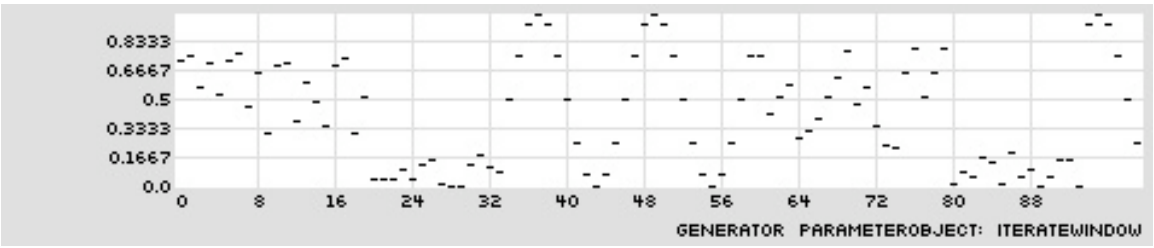
- BasketSelect: Select values form a list using another PO providing values within the unit interval

```
:: tpmap 100 bs,(-3,-2,-
1,0,1,2,3),(ru,(bpl,e,l,((0,.5),(100,1))),(bpl,e,l,((0,.5),(100,0))))
basketSelect, (-3,-2,-1,0,1,2,3), (randomUniform, (breakPointLinear, event,
loop, ((0,0.5),(100,1))), (breakPointLinear, event, loop, ((0,0.5),(100,0)))),
TPmap display complete.
```



- IterateWindow: Select from a list of POs, and then draw a selected number of values from that PO

```
:: tpmap 100 iw,((ru,.2,.8),(re,15,0,1),(ws,e,12,0,0,1)),(bg,rp,(14,20,26)),oc
iterateWindow, ((randomUniform, (constant, 0.2), (constant, 0.8)),
(randomExponential, 15.0, (constant, 0), (constant, 1)), (waveSine, event,
(constant, 12), 0, (constant, 0), (constant, 1))), (basketGen, randomPermutate,
(14,20,26)), orderedCyclic
TPmap display complete.
```



Chapter 8. Meeting 8, Approaches: Permutations, Generators, and Chaos

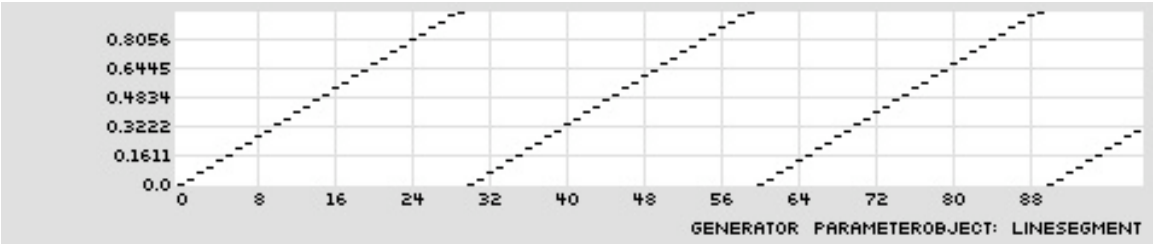
8.1. Announcements

- KIOKU concert this Friday, 6:30, in the MIT Lewis Music Library
- Musical Design Report 2 due 11 March: details to follow
- Sonic System Project Draft due 27 April: start thinking

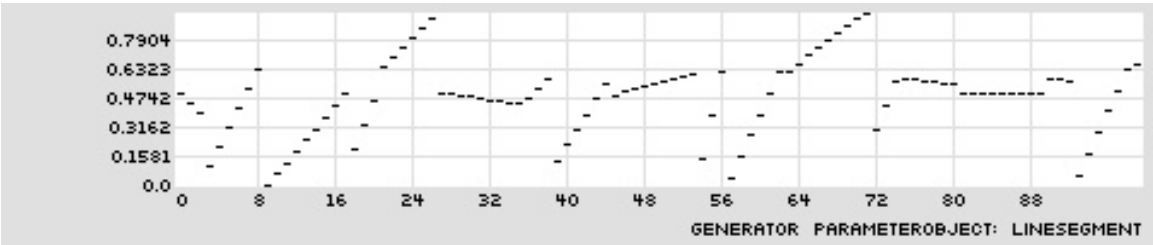
8.2. A Line Segment Generator

- Often we need to vary a parameter linearly over time or events
- Break point functions require defining individual points
- LineSegment: A dynamic line generator between minimum and maximum values over a duration

```
:: tpmap 100 ls,e,30,0,1
lineSegment, (constant, 30), (constant, 0), (constant, 1)
TPmap display complete.
```



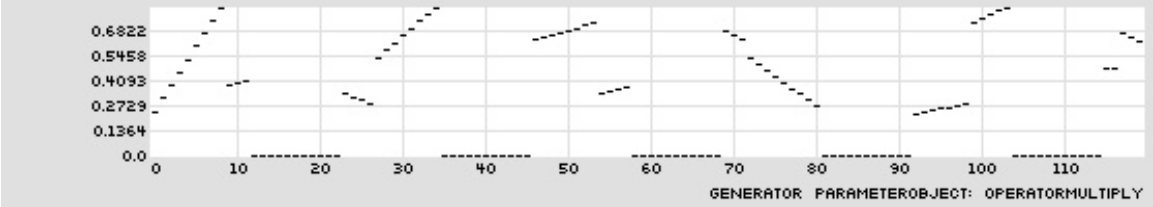

```
:: tpmap 100 ls,e,(bg,oc,(3,6,9)),(ru,0,.7),(ru,.3,1)
lineSegment, (basketGen, orderedCyclic, (3,6,9)), (randomUniform, (constant, 0),
(constant, 0.7)), (randomUniform, (constant, 0.3), (constant, 1))
TPmap display complete.
```



8.3. Large Scale Amplitude Behavior with Operators

- By multiplying or summing multiple behaviors, dynamic large-scale shapes are possible
- Multiplying amplitudes by zero can create periods of inactivity
- Techniques derived from modular synthesis
- OperatorMultiply used to scale LineSegment and WavePulse

```
:: tpmap 120 om,(ls,e,9,(ru,.2,1),(ru,.2,1)),(wp,e,23,0,0,1)
operatorMultiply, (lineSegment, (constant, 9), (randomUniform, (constant, 0.2),
(constant, 1)), (randomUniform, (constant, 0.2), (constant, 1))), (wavePulse,
event, (constant, 23), 0, (constant, 0), (constant, 1))
TPmap display complete.
```



- Command sequence:
  - emo mp
  - tin a 64
  - tie r pt,(bg,rp,(16,16,8)),(bg,rp,(2,2,1,4)),(c,1)
  - tie a om,(ls,e,9,(ru,.2,1),(ru,.2,1)),(wp,e,23,0,0,1)
  - eln; elh
- Related ParameterObjects: OperatorAdd, OperatorMultiply, OperatorDivide, OperatorPower, OperatorSubtract, OperatorCongruence

8.4. Reading: Ames. A Catalog of Sequence Generators: Accounting for Proximity, Pattern, Exclusion, Balance and/or Randomness

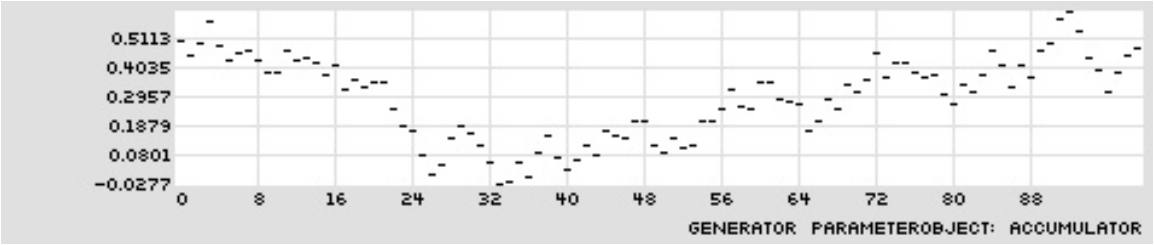
- Ames, C. 1992. "A Catalog of Sequence Generators: Accounting for Proximity, Pattern, Exclusion, Balance and/or Randomness." *Leonardo Music Journal* 2(1): 55-72.
- What does ames mean by dependence, exclusion, and balance

- Why does Ames have so many varieties of random uniform generators, such as LEHMER, SPREAD, FILL, and others?
- How is Brownian motion related to random walks?
- How does Ames characterize the artistic opportunities of using 1/f noise?
- What are the characteristics of output provided by chaotic generators such as LOGISTIC and BAKER
- What is the idea of a chaos knob?

8.5. Continuous Random Walks

- We can use BasketGen for discrete random walks
- We can use Accumulator for continuous random walks
- Accumulator

```
:: tpmap 100 a,.5,(ru,-.1,.1)
accumulator, 0.5, (randomUniform, (constant, -0.1), (constant, 0.1))
TPmap display complete.
```



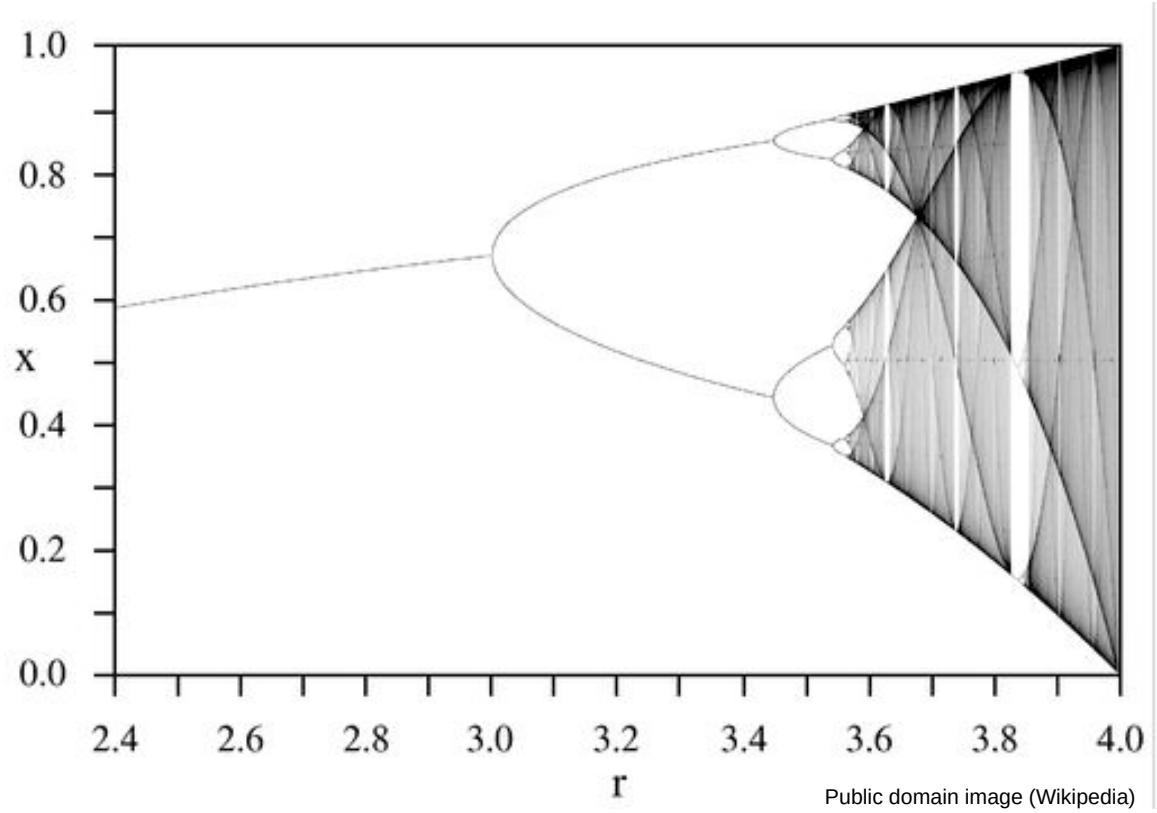
8.6. Chaos and the Logistic Map

- Complex dynamical systems
- Deterministic systems that exhibit complex behavior
- Most employ iterative processing and result in sensitivity to initial conditions (butterfly effect)
- The logistic map was developed as a model of population growth by Pierre Verhulst

  $x_{n+1} = rx_n(1 - x_n)$

  r is a positive number between 0 and 4 that represents a combined rate for reproduction and starvation

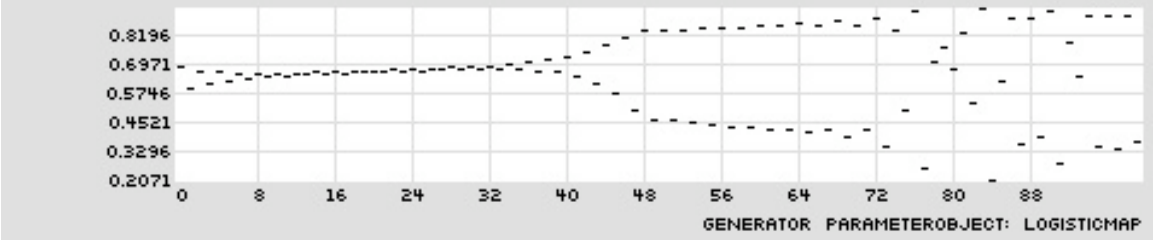
- States produces constant outputs, oscillating behavior, and complex behavior



Public domain image (Wikipedia)

- LogisticMap: most interesting output available from *p* (or r, lambda, or chaos knob) between 2.75 and 4

```
:: tpmap 100 lm,.5,(ls,e,100,2.75,4),0,1
logisticMap, 0.5, (lineSegment, (constant, 100), (constant, 2.75), (constant,
4)), (constant, 0), (constant, 1)
TPmap display complete.
```



- Related ParameterObjects: henonBasket, lorenzBasket

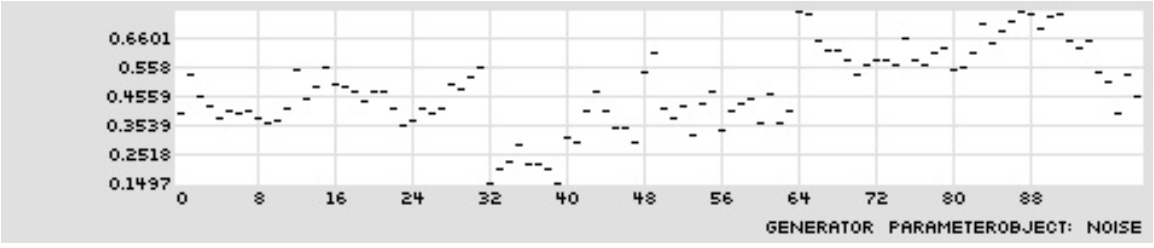
8.7. Reading: Voss and Clarke. 1/f Noise in Music: Music from 1/f Noise

- Voss, R. F. and J. Clarke. 1978. "1/f Noise in Music: Music from 1/f Noise." *Journal of the Acoustical Society of America* 63(1): 258-263.
- What is a 1/f noise, and what is the variations of noise from 1/f0, 1/f1, 1/f2, 1/f3?
- The sound and shape of correlated noise: [noiseColors.pd]
- What technique did Voss and Clarke use to analyze music?
- What sort of data did Voss and Clarke collect?
- Extracting an amplitude envelope from an audio signal: [vossClarke.pd]
- What conclusions do Voss and Clarke make about 1/f spectral densities?
- Is music (or the averaged spectral analysis of amplitude envelopes) intelligent behavior
- What technique did Voss and Clarke use to generate melodies? Is this technique parallel to the analysis technique?
- What conclusions did they draw from human evaluation of their generated melodies? Were these conclusions based on the 1/f noise source?

8.8. 1/f Noise

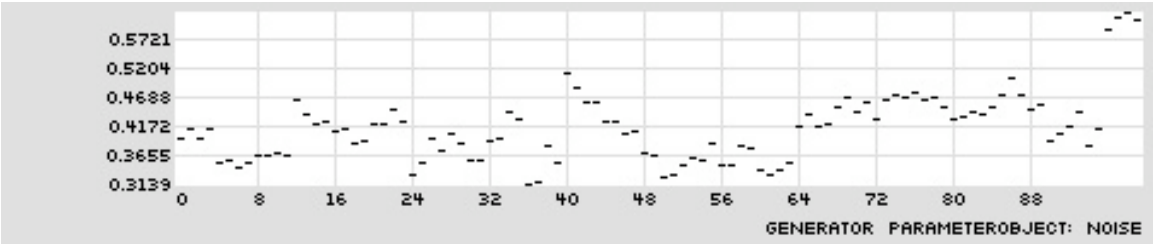
- Rather than just one type of 1/f noise, use many
- Gamma can move between 0 (white), 1 (pink), 2 (brown), 3 (black)
- 1/f noise: gamma == 1

```
:: tpmap 100 n,100,1,0,1
noise, 100, (constant, 1), (constant, 0), (constant, 1)
TPmap display complete.
```



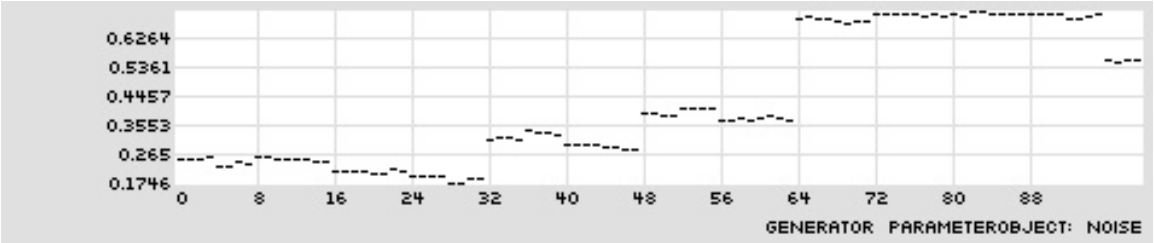
- 1/f noise: gamma == 2

```
:: tpmap 100 n,100,2,0,1
noise, 100, (constant, 2), (constant, 0), (constant, 1)
TPmap display complete.
```



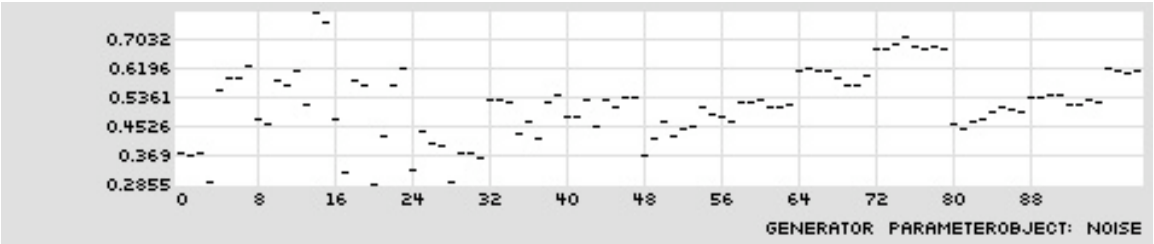
- 1/f noise: gamma == 3

```
:: tpmap 100 n,100,3,0,1
noise, 100, (constant, 3), (constant, 0), (constant, 1)
TPmap display complete.
```



- Noise: dynamically varying the gamma

```
:: tpmap 100 n,100,(ls,e,100,0,3),0,1
noise, 100, (lineSegment, (constant, 100), (constant, 0), (constant, 3)),
(constant, 0), (constant, 1)
TPmap display complete.
```

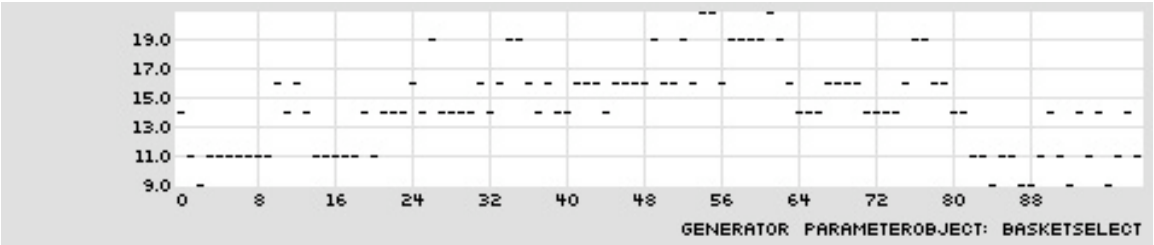


8.9. 1/f Noise in Melodic Generation: LineGroove

- Using BasketSelect to select discrete values from a continuous generator

```
:: tpmap 100 bs,(2,4,7,9,11,14,16,19,21,23),(n,100,1,0,1)
basketSelect, (2,4,7,9,11,14,16,19,21,23), (noise, 100, (constant, 1),
```

```
(constant, 0), (constant, 1)),
TPmap display complete.
```



- Command sequence using TM LineGroove:
  - emo m
  - tmo lg
  - tin a 108
  - tie r cs,(ls,e,10,(ru,.01,.2),(ru,.01,.2))
  - tie f bs,(2,4,7,9,11,14,16,19,21,23),(n,100,1,0,1)
  - eln; elh

8.10. 1/f Noise in Melodic Generation: HarmonicAssembly

- Command sequence using TM Harmonic Assembly:
  - emo m
  - pin a d3,e3,g3,a3,b3,d4,e4,g4,a4,b4,d5,e5,g5,a5,b5
  - tmo ha
  - tin a 27
  - tie r pt,(c,16),(ig,(bg,rc,(1,2,3,5,7)),(bg,rc,(3,6,9,12))),(c,1)
  - tie a om,(ls,e,9,(ru,.2,1),(ru,.2,1)),(wp,e,23,0,0,1)
  - tie d0 c,0
  - tie d1 n,100,2,0,14
  - tie d2 c,1
  - tie d3 c,1

- eln; elh

- Continued command sequence: with chord size generation between 1 through 4
  - tie d3 ru,1,4
  - eln; elh

8.11. Tutorial: PD Arrays as Parameters: Filtered Noise

- [mgEnvlMtAr] creates a mono-triggered, attack-release envelope

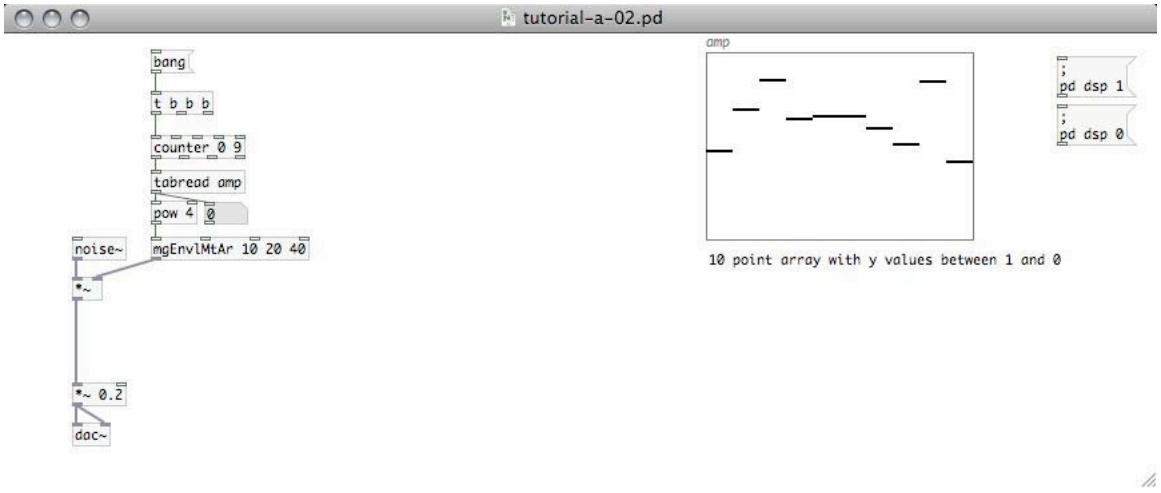
  [mgEnvlMtAr] arguments: attack time, release time, duration

  [mgEnvlMtAr] trigger: a floating point value that sets the peak amp

8.12. Tutorial: PD Arrays as Parameters: Cyclical Amplitude Values

- Looping through an array with amplitude values with [counter]

  [pow 4] provides non-linear to linear amplitude scaling



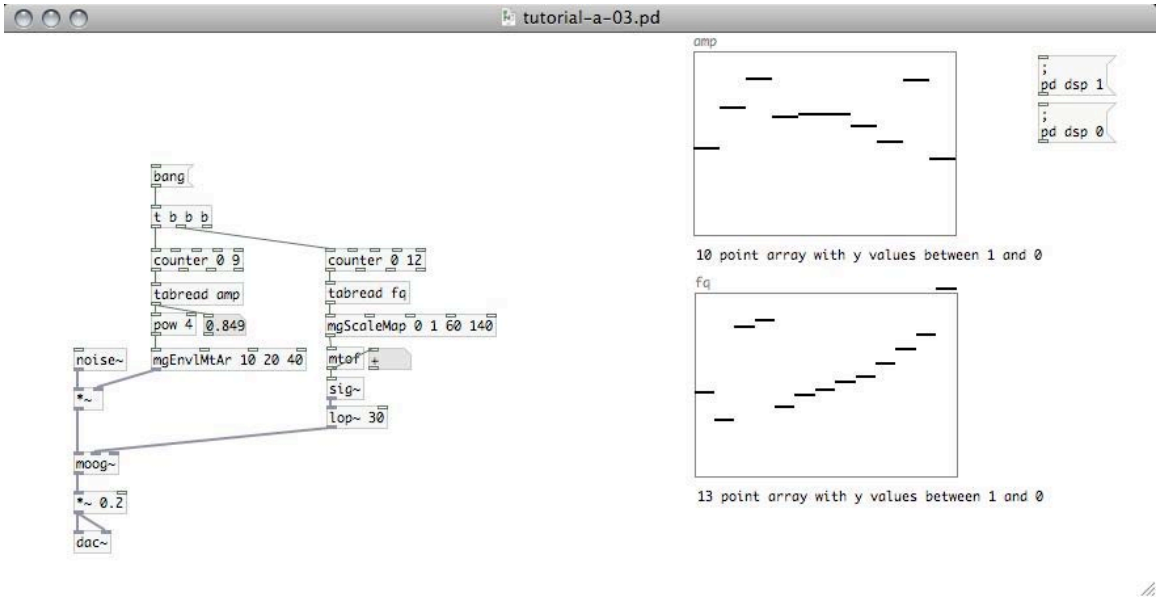
8.13. Tutorial: PD Arrays as Parameters: Cyclical Cutoff Frequency Values

- Looping through an array if values scaled to MIDI pitch values (60-140) with [mgScaleMap]

  MIDI pitch values are scaled to frequency values with [mtof]

  Data values are converted to signals with [mtof] and [lop~ 30]

  [moog~] provides a signal controlled low pass filter with variable resonance



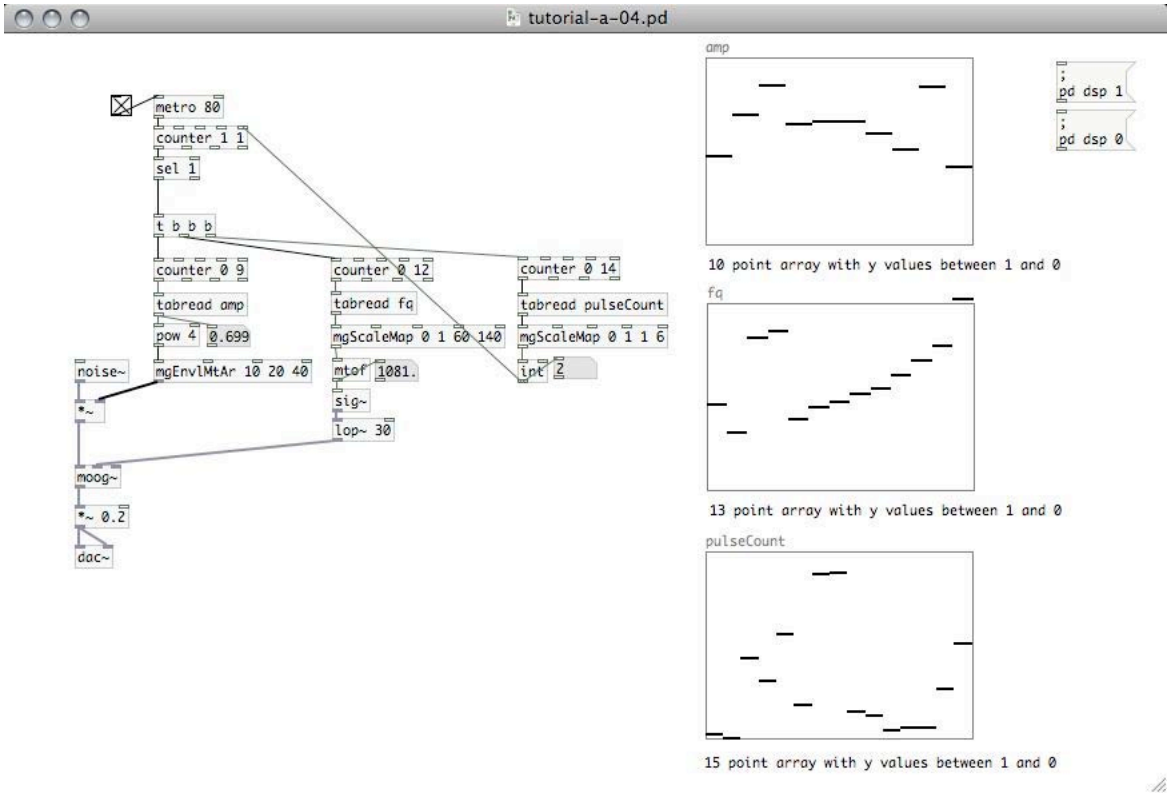
8.14. Tutorial: PD Arrays as Parameters: Cyclical Pulse Multipliers

- [metro] provides regularly spaced triggers

  [counter] and [sel 1] are used to select the first of each cycle

  An array of values is scaled to pulse multipliers with [mgScaleMap]

  The [counter] max value is dynamically set after reading and mapping a value from the array



Chapter 9. Meeting 9, History: Lejaren Hiller

9.1. Announcements

- Musical Design Report 2 due 11 March: details to follow
- Sonic System Project Draft due 27 April: start thinking

9.2. Musical Design Report 2

- May be primarily rhythmic or melodic, or neither
- Must have, in at least one section, 6 active timbre sources
- Must have, in at least one section, a feeling of time without regular pulse
- Should have at least an AB or ABA form
- Must feature 1/f noise and Markov-chains in some manner
- Can be composed with athenaCL, athenaCL and other tools, or other tools alone

9.3. Chronology: Early Experiments in Algorithmic Composition with a Computer

- late 1955: Caplin and Prinz: Mozart Contradance Dice Game
- July 1956: Klein and Bolitho: Push Button Bertha
- August (movement 1) and November (complete) 1956: Hiller and Isaacson: Illiac Suite
- 1964, 1969: Koenig's PR1 and PR2

9.4. Hiller and Isaacson

- Lejaren Hiller (1924-): research chemist for du Pont, worked at University of Illinois, explored applications of computers to chemical problems; studied music theory and composition after Illiac work

  Isaacson (1930-): applications of computers to chemical problems, worked for Standard Oil in California; no musical training

Photo of L. A. Hiller and L. M. Isaacson removed due to copyright restrictions.

- Used University of Illinois ILLIAC (ILLInois Automated Computer)

  1952: ILLIAC, the first computer built and owned entirely by an educational institution

- Created four movements of a string quartet: *Illiac Suite*
- Hiller describes the *Illiac Suite* as a "… presentation of sample results … in the form of a four-movement transcription for string quartet" (1956, p. 248).
- Published a complete book on the process: Hiller, L. and L. Isaacson. 1959. *Experimental Music.* New York: McGraw-Hill.
- Hiller went to continue to explore techniques of computer composition, including work with John Cage

9.5. Reading: Hiller, L. and L. Isaacson. Musical Composition with a High-Speed Digital Computer

- Hiller, L. and L. Isaacson. 1958. "Musical Composition with a High-Speed Digital Computer." *Journal of the Audio Engineering Society* 6(3): 154-160.

- Why do HIller and Isaacson think that music is well suited for this sort of computer experiment?
- Did Hiller and Isaacson see their work as an experiment, or as a work of art?
- What social and critical context is suggested by the discussion question, at the end of the article?

9.6. Hiller and Isaacson: Illiac Suite I and II

- Strict counterpoint in the model of 18th century treatise *Gradus ad Parnassum*
- Monte-carlo technique: random generative pitches and filter through rules
- Borrowed programming models from previous work in chemistry
- Generated only pitch; registration, instrumentation, dynamics, and rhythm manually applied
- Flow chart of strict counterpoint

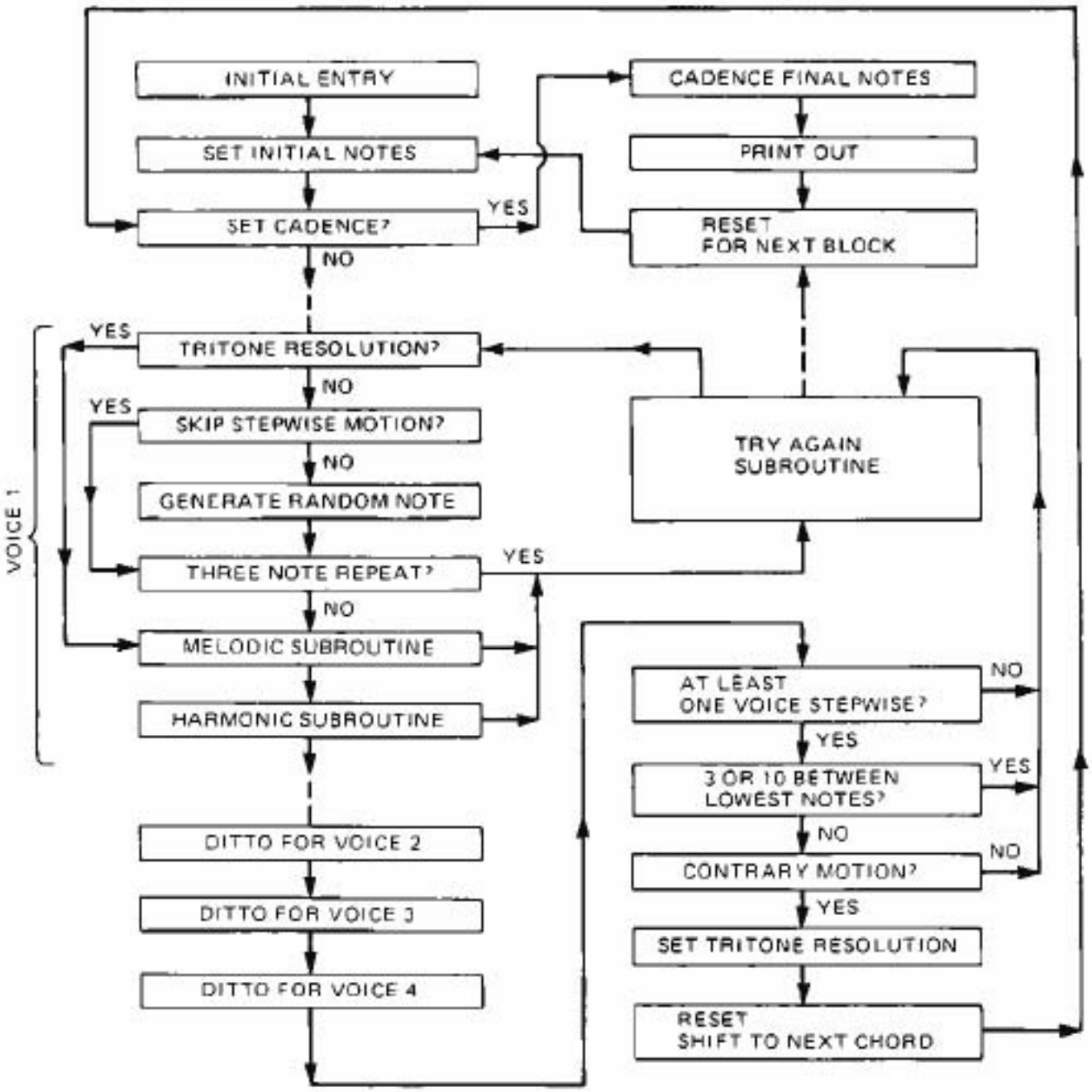


Figure 1.1
Experiment 2: Main routine for strict counterpoint.

Audio: Hiller: Illiac Suite, Experiment 1 and Experiment 2 (1956)

9.7. Monte Carlo: Concepts

- Monte-Carlo: a wealthy quarter of the city-state Principality of Monaco, and host to European Formula One racing, resorts, and gambling
- 1940s: John von Neuman and Stanislas Ulam: used to study problems of neutron diffusion at Los Alamos in research relating to the hydrogen bomb
- Random generation of values that are tested and then kept or discarded
- Only feasible with the use of computers
- Brute-force solutions
- Good for problems where attributes of the answer are known, but how to get the answer is not
- Also called statistical sampling; related to constraint satisfaction problems

9.8. Monte Carlo Melodic Generation with athenaCL Python Libraries

- Produce a melody using 14 diatonic pitches, where intervals between steps are limited between two values provided with command-line arguments
- monteCarlo.py

```
import os, random, sys
from athenaCL.libATH import midiTools
from athenaCL.libATH import osTools
from athenaCL.libATH import pitchTools
from athenaCL.libATH import rhythm
from athenaCL.libATH.libOrc import generalMidi
from athenaCL.libATH.libPmtr import parameter

OUTDIR = '/Volumes/xdisc/_scratch'
BEATDUR = rhythm.bpmToBeatTime(128) # provide bpm value

def getInstName(nameMatch):
   for name, pgm in generalMidi.gmProgramNames.items():
      if name.lower().startswith(nameMatch.lower()):
         return pgm # an integer
   return None

def convertPitch(pitch, octShift=0):
   midiPs = pitchTools.psToMidi(pitchTools.psNameToPs(pitch))
   midiPs = midiPs + (12*octShift)
   return midiPs

def genScore(minStep=1, maxStep=3):
   pitchScale = {1:'C4', 2:'D4', 3:'E4', 4:'F4', 5:'G4', 6:'A4', 7:'B4',
                 8:'C5', 9:'D5',10:'E5',11:'F5',12:'G5',13:'A5',14:'B5',
                }
   melodyLength = 36
   melody = []
   while True:
      if len(melody) == melodyLength:
```

```
            break
         elif len(melody) == 0:
            melody.append(1)
            continue
         else:
            pitchLast = melody[-1]
            while True:
               pitchNew = random.choice(pitchScale.keys())
               interval = abs(pitchNew - pitchLast)
               if interval >= minStep and interval <= maxStep:
                  melody.append(pitchNew)
                  break
               else:
                  continue
      score = []
      tStart = 0.0
      for i in range(melodyLength):
         pitch = convertPitch(pitchScale[melody[i]])
         dur = BEATDUR * .5
         amp = 90
         pan = 63
         event = [tStart, dur, amp, pitch, pan]
         score.append(event)
         tStart = tStart + dur
      return score

   def main(minStep, maxStep):
      trackList = []
      score = genScore(minStep, maxStep)
      trackList.append(['part-a',  getInstName('piano'), None, score])
      path = os.path.join(OUTDIR, 'test.midi')
      mObj = midiTools.MidiScore(trackList)
      mObj.write(path)
      osTools.openMedia(path)

   if __name__ == '__main__':
      if len(sys.argv) != 3:
         print('required command-line arguments: minStep maxStep')
      else:
         main(int(sys.argv[1]), int(sys.argv[2]))
```

9.9. Hiller and Isaacson: Illiac Suite III

- Constrained chromatic music
- Generated pitch, rhythm, amplitude, and performance articulation
- Audio: Hiller: Illiac Suite, Experiment 3 (1956)

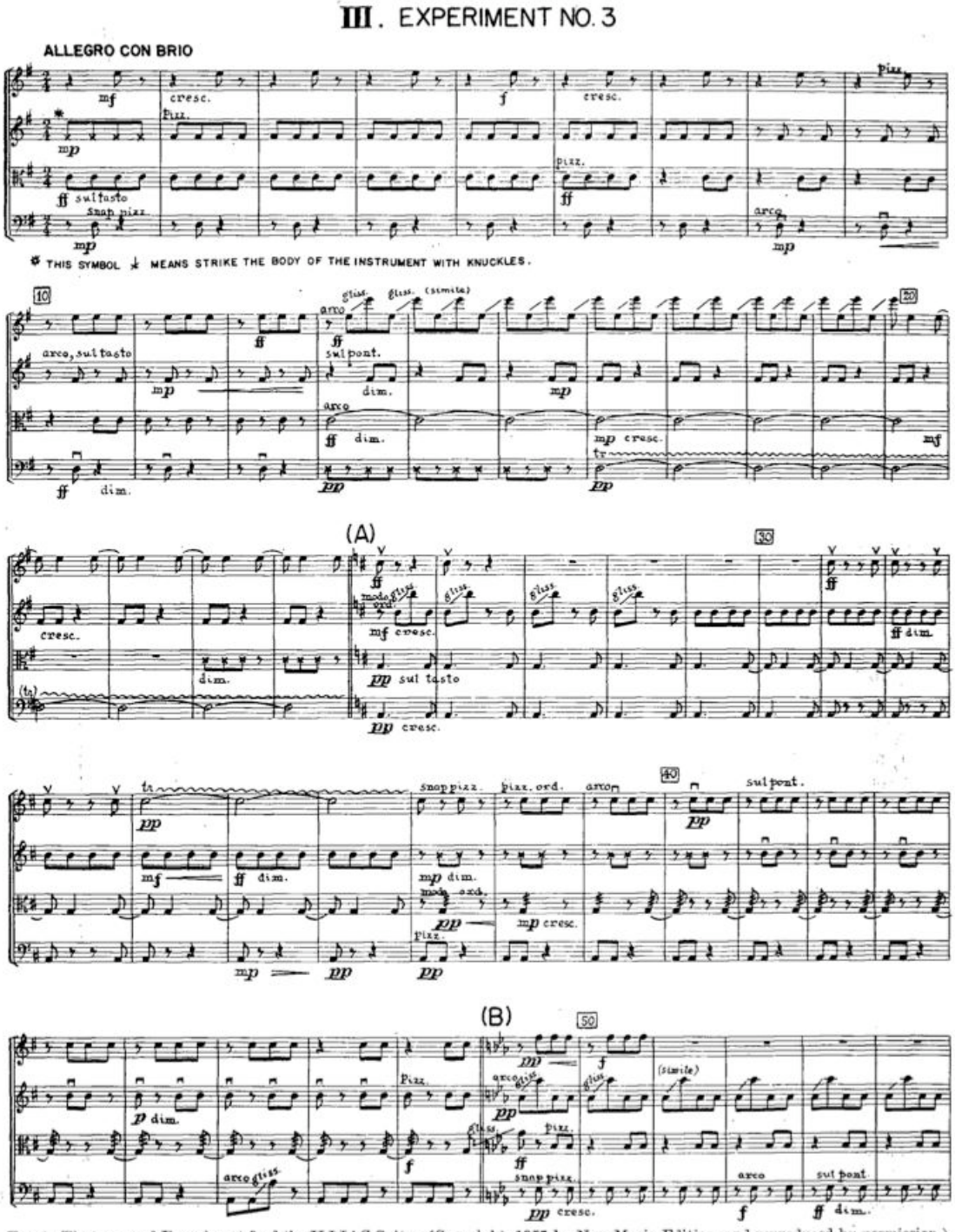


FIG. 3. First page of Experiment 3 of the ILLIAC Suite. (Copyright, 1957 by New Music Edition, and reproduced by permission.)

9.10. Hiller and Isaacson: Illiac Suite IV

- Markov chains (zero and first order) for interval and harmony selection

- Models from music theory (Schenker)
- Only movement not produced from a combination of outputs
- Tempo, meter, dynamics added manually
- Audio: Hiller: Illiac Suite, Experiment 4 (1956)

9.11. Hiller and Isaacson: Issues and Responses

- Cony, E. 1956. "Canny Computers: Machines Write Music, Play Checkers, Tackle New Tasks in Industry." *Wall Street Journal* 148(56)

Canny Computers

Machines Write Music, Play Checkers, Tackle New Tasks in Industry

But Some Executives Still Shy From Computers; Manpower Remains Scarce

Boost for Baseball Bettors

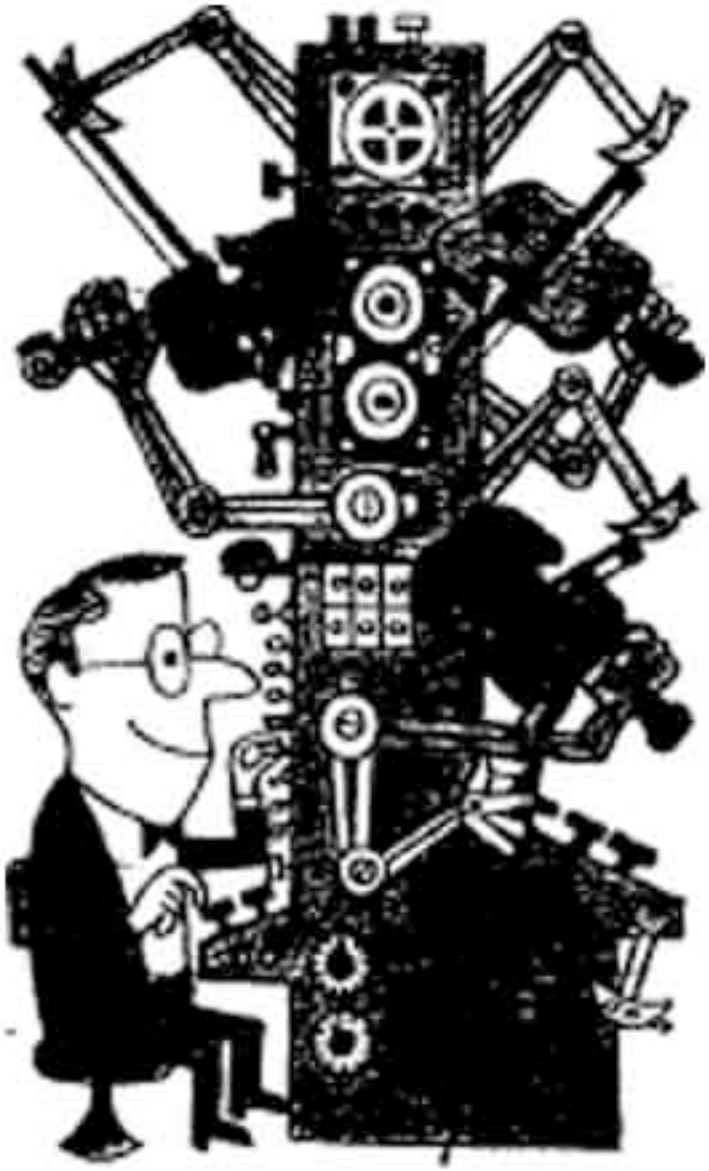
By Ed Cony
Staff Reporter of The Wall Street Journal

LOS ANGELES—Picking baseball winners and planning aircraft production. Playing checkers and checking up on guided missiles. Composing music and forecasting the weather.

These are only a few of the widely diverse tasks now being performed by those highly-versatile machines known as electronic computers. Computer experts daily are discovering new jobs for the complex machines. And they're convinced the business world has only begun to tap the computer's potentialities.

"I'd say 99% of the firms using computers aren't getting the most out of them, because they insist on using them for bookkeeping and accounting operations," says Norman J. Ream, Lockheed Aircraft Corp.'s director of systems planning.

- Brower, B. 1961. "Why 'Thinking Machines' Cannot Think." *New York Times* February 19: 213.



QUADRATIC QUARTET — The latest manifestation is a machine-composed work, "Illiac Suite for String Quartet."

- "And finally -- to stretch the point as far as some of the computer people have done -- machines are presumably capable of 'creating works of art.' in any case, Lejaren A Hiller Jr. and L. M. Isaacson hold a copyright for their 'Illiac Suite for String Quartet' ..."

"this rather ludicrous extension of the machine-brain equation to artistic creativity perhaps best illustrates its limitations. No machine is every really likely to contain the artist within its electro-physics, and to a greater or lesser degree, it is unlikely that machine equivalents will be constructed for the highest of human attributes."

"it is best to view the electronic brains as instruments of human calculation, which achieve results that lie beyond human time and precision, but not beyond human intelligence"

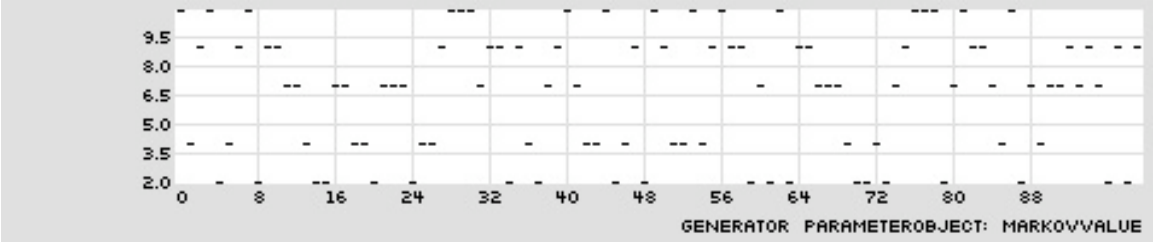
9.12. Zero Order Markov Chains as ParameterObjects

- A zero order Markov chain is weighted random selection
- MarkovValue ParameterObject

```
:: tpv mv
Generator ParameterObject
{name,documentation}
MarkovValue         markovValue, transitionString, parameterObject
                    Description: Produces values by means of a Markov transition
                    string specification and a dynamic transition order
                    generator. Markov transition order is specified by a
                    ParameterObject that produces values between 0 and the
                    maximum order available in the Markov transition string. If
                    generated-orders are greater than those available, the
                    largest available transition order will be used. Floating-
                    point order values are treated as probabilistic weightings:
                    for example, a transition of 1.5 offers equal probability of
                    first or second order selection. Arguments: (1) name, (2)
                    transitionString, (3) parameterObject {order value}
```

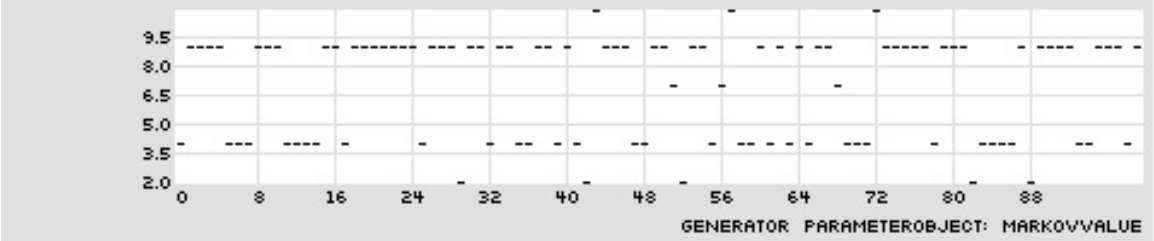
- The transition string
  - Two parts: symbol definitions and weights
  - Symbol definition: a{3}b{345}c{23.54}
  - Zero order weights: :{a=3|b=1|c=34}
- MarkovValue: zero order with equal weighting

```
:: tpmap  100 mv,a{2}b{4}c{7}d{9}e{11}:{a=1|b=1|c=1|d=1|e=1}
markovValue, a{2}b{4}c{7}d{9}e{11}:{a=1|b=1|c=1|d=1|e=1}, (constant, 0)
TPmap display complete.
```



- MarkovValue: zero order with stronger weightings on two values

```
:: tpmap  100 mv,a{2}b{4}c{7}d{9}e{11}:{a=1|b=6|c=1|d=9|e=1}
markovValue, a{2}b{4}c{7}d{9}e{11}:{a=1|b=6|c=1|d=9|e=1}, (constant, 0)
TPmap display complete.
```



9.13. Building a Self-Similar Melody

- Self similar Markovian melody generation and transposition
- Command sequence:
  - emo m
  - tin a 24
  - *using 1/f noise for durations with ConvertSecond and Noise*

    tie r cs,(n,100,1.5,.100,.180)
  - *a more dynamic timing offset*

    tie r cs,(om,(n,100,1.5,.100,.180),(ws,t,8,0,.5,1))
  - *Markov weighted pitch transposition*

    tie f mv,a{2}b{4}c{7}d{9}e{11}:{a=1|b=6|c=1|d=9|e=1}
  - *self-similar pitch transposition combing a grouped version of the same Markov generator with OperatorAdd*

    tie f oa,(mv,a{2}b{4}c{7}d{9}e{11}:{a=1|b=3|c=1|d=3|e=1}),
    (ig,(mv,a{2}b{4}c{7}d{9}e{11}:{a=1|b=3|c=1|d=3|e=1}),(ru,10,20))
  - *Markov based octave shifting*

    tie o mv,a{-2}b{0}c{-2}d{0}e{-1}:{a=1|b=3|c=1|d=3|e=1}
  - *A widening beta distribution*

    tie a rb,.2,.5,(ls,e,(ru,3,20),.5,1)

- *Modulated with a pulse wave (and random frequency modulation on the PulseWave)*

  tie a om,(rb,.2,.5,(ls,e,(ru,3,20),.5,1)),(wp,e,(ru,25,30),0,0,1)

- tie t 0,120
- eln; elh

9.14. Resuming PD Tutorial

- PD Tutorial

Chapter 10. Meeting 10, Approaches: Probability and Markov Chains

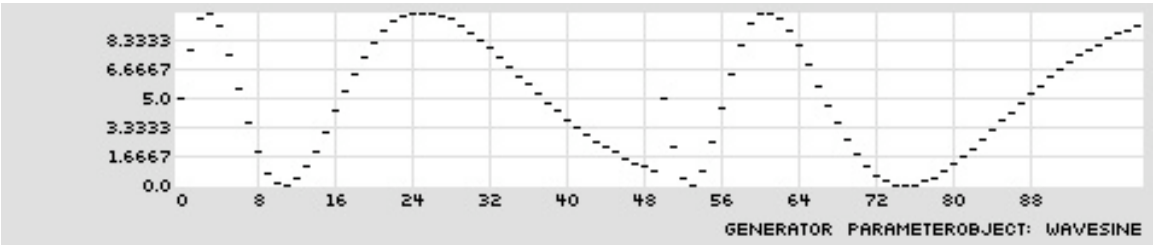
10.1. Announcements

- Musical Design Report 2 due this Thursday, 11 March
- Thursday we will work in PD and Csound
- Quiz next Tuesday

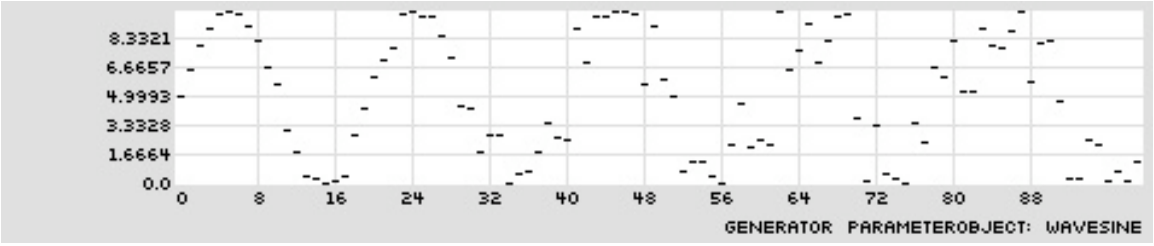
10.2. Half-Period Oscillators as ParameterObjects

- Continuously varying the seconds per cycle (frequency) of an oscillator results in complex periodicities; random or discrete frequency variation results in complexity

```
:: tpmap 100 ws,e,(ls,e,50,10,30),0,0,10
waveSine, event, (lineSegment, (constant, 50), (constant, 10), (constant, 30)),
0, (constant, 0), (constant, 10)
TPmap display complete.
```



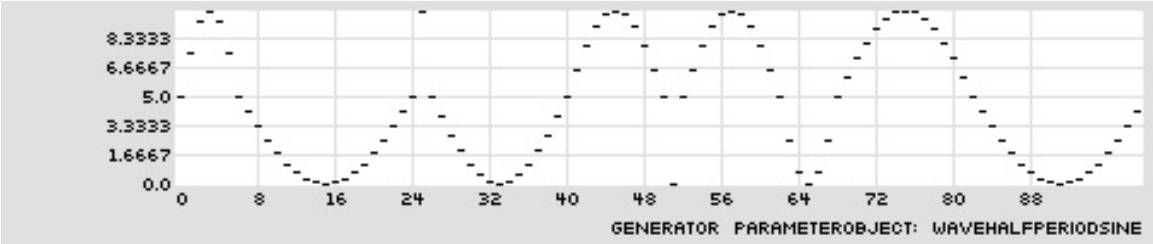

```
:: tpmap 100 ws,e,(ru,19,21),0,0,10
waveSine, event, (randomUniform, (constant, 19), (constant, 21)), 0, (constant,
0), (constant, 10)
TPmap display complete.
```



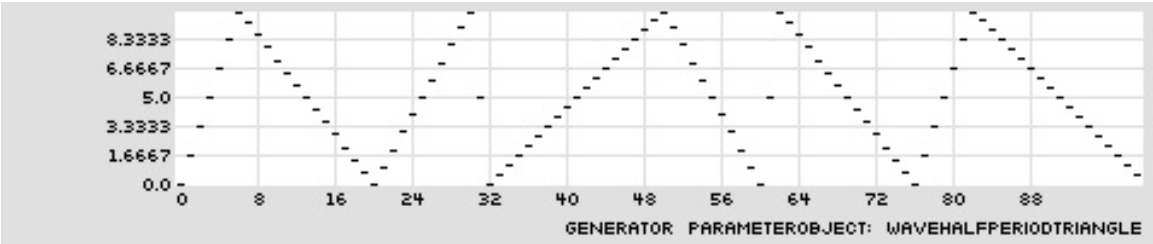
- An alternative is an oscillator that only updates seconds per half cycle (half frequency) once per half-period

WaveHalfPeriodSine, WaveHalfPeriodTriangle, WaveHalfPeriodPulse, WaveHalfPeriodCosine

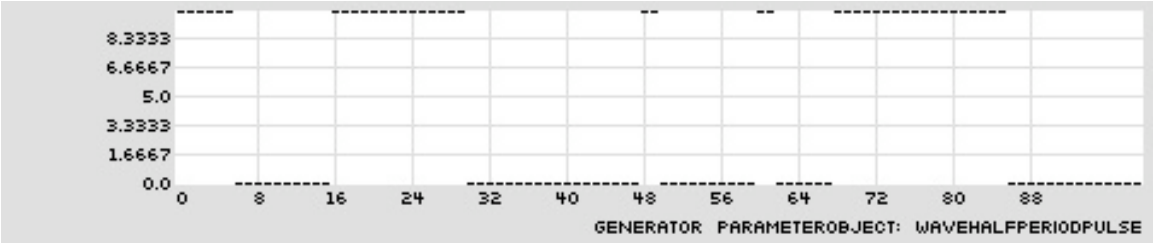
```
:: tpmap 100 whps,e,(bg,rp,(2,6,10,14,18)),0,0,10
waveHalfPeriodSine, event, (basketGen, randomPermutate, (2,6,10,14,18)), 0,
(constant, 0), (constant, 10)
TPmap display complete.
```




```
:: tpmap 100 whpt,e,(bg,rp,(2,6,10,14,18)),0,0,10
waveHalfPeriodTriangle, event, (basketGen, randomPermutate, (2,6,10,14,18)), 0,
(constant, 0), (constant, 10)
TPmap display complete.
```




```
:: tpmap 100 whpp,e,(bg,rp,(2,6,10,14,18)),0,0,10
waveHalfPeriodPulse, event, (basketGen, randomPermutate, (2,6,10,14,18)), 0,
(constant, 0), (constant, 10)
TPmap display complete.
```



10.3. Markov Analysis and Generation: Basics

- Examine an ordered sequence states
- Given an event at *n*-1, what is the probability of any state (of all possible states) at *n*?

- Look at all possible *n*-1 states, and find how often they move to each state at *n*
- Use these probabilities to re-generate new sequences (where more frequent states result in proportionally weighted randomness)

10.4. Markov Analysis and Generation: Orders

- Zeroth order: examine 0 past states; given all possible states, generate *n* based on the distribution of all states.
- First order: examine 1 past state; generate *n* based on the probability of *n*-1 moving to each state.
- Second order: examine 2 past states; generate *n* based on the probability of *n*-2 and *n*-1 moving to each state.
- Second order: examine 3 past states; generate *n* based on the probability of *n*-3, *n*-2 and *n*-1 moving to each state.
- The greater the order, the more the past is taken into account in determining the next state
- The greater the order, the more the output is similar to the source

10.5. Reading: Ames: The Markov Process as a Compositional Model: A Survey and Tutorial

- Ames, C. 1989. "The Markov Process as a Compositional Model: A Survey and Tutorial." *Leonardo* 22(2): 175-187.
- What does Ames refer to by stationary probabilities
- What does Ames claim as the greatest strength of Markov chains?
- What technique does Ames suggests as a way to create large-scale behavior out of Markov chains?

10.6. Markov Chains: History

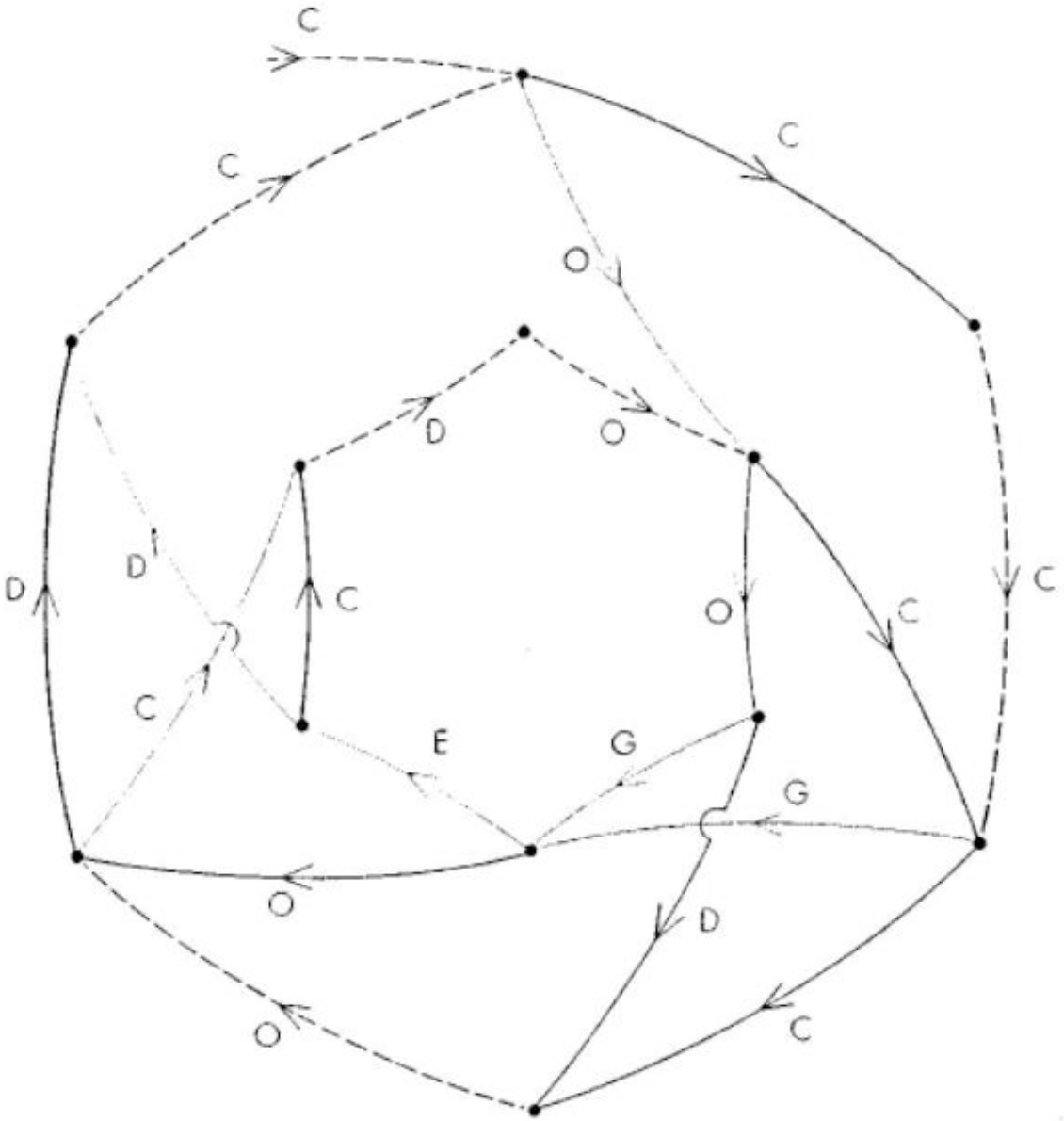
- 1906: Andrey Andreyevich Markov, Russian mathematician

  Used Markov chains to show tendencies in written Russian in a text by Pushkin

- 1949: Claude E. Shannon and Warren Weaver: *A Mathematical Theory of Communication*; associated with information theory
  - Demonstrate using stochastic processes to generate English sentences
  - Suggest application to any sequence of symbols, including music

10.7. Markov Chains: History: Early Musical Applications

- The "Banal Tune-Maker" of Richard C. Pinkerton (1956)



BANAL TUNE-MAKER produces simple, redundant melodies that sound like nursery tunes. A sequence of notes is obtained by following a path through the network, starting at the top, and writing down the note (or rest) attached to each segment traversed. Where there is a choice of paths, a coin is flipped. If it comes up heads, the black path is taken; if tails, the colored path. Broken lines show the path from a junction where there is no choice.

	O	C	D	E	F	G	A	B
O	0.38	0.17	0.10	0.10	0.06	0.13	0.03	0.02
C	0.36	0.23	0.13	0.07	0.02	0.10	0.03	0.07
D	0.26	0.20	0.21	0.19	0.03	0.06	0.01	0.05
E	0.22	0.15	0.18	0.16	0.16	0.12	0.01	0.00
F	0.15	0.00	0.14	0.35	0.14	0.20	0.01	0.01
G	0.29	0.14	0.00	0.16	0.06	0.26	0.08	0.00
A	0.17	0.05	0.07	0.00	0.02	0.36	0.15	0.17
B	0.18	0.30	0.12	0.01	0.01	0.08	0.21	0.08

TRANSITION PROBABILITIES show how frequently any note follows any other in the 39 nursery tunes. The first notes of all possible pairs are listed in the column at the left; the second notes, in the row at the top. Thus each number in the table gives the probability that the note at the top of its column will come after the note at the left of its row. The color pattern divides the table between likely transitions (*colored*) and unlikely (*white*).

- John F. Sowa with a Geniac "Electronic Brain Kit" (1957)

BUILD 125 COMPUTERS AT HOME WITH GENIAC®

ONLY **$19.95**

With the 1958 model **GENIAC®**, original electric brain construction kit, seven books and pamphlets, 400 parts and components, all materials for experimental computer lab plus **DESIGN-O-Mat®**.

A COMPLETE COURSE IN COMPUTER FUNDAMENTALS

The **GENIAC** Kit is a complete course in computer fundamentals, in use by thousands of colleges, schools and private individuals. Includes everything necessary for building an astonishing variety of computers that reason, calculate, solve codes and puzzles, forecast the weather, compose music, etc. Included in every set are five books described below, which introduce you step-by-step to the wonder and variety of computer fundamentals and the special problems involved in designing and building your own experimental computers.

Build any one of these 125 exciting electric brain machines in just a few hours by following the clear step-by-step directions given in these books. No soldering. . . . **GENIAC** is a genuine electric brain machine—not a toy. The only logic and reasoning machine kit in the world that not only adds and subtracts but presents the basic ideas of cybernetics, boolean algebra, symbolic logic, automation, etc. So simple to construct that a twelve-year-old can build what will fascinate a Ph.D. You can build machines that compose music, forecast the weather.

TEXT PREPARED BY MIT SPECIALIST

Dr. Claude Shannon, a research mathematician for Bell Telephone Laboratories, a research associate at MIT. His books include Communication theory and the recent volume "Automation Studies" on the theory of robot construction. He has prepared a paper entitled "A Symbolic Analysis of Relay and Switching Circuits" available in the **GENIAC**. Covers basic theory necessary for advanced circuit design, it vastly extends the range of our kit.

The complete design of the kit and the manual as well as the special book **DESIGN-O-Mat®** was co-created by Oliver Garfield, author of "Minds and Machines," editor of the "Gifted Child Magazine" and the "Review of Technical Publications."

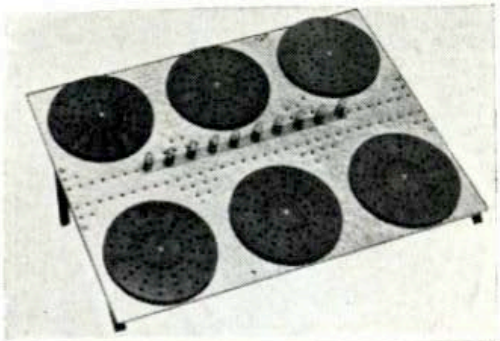
Oliver Garfield Co., Inc. Dept. ASF-108

108 East 16th St., N. Y. 3, N. Y.

Please send me at once the **GENIAC** Electric Brain Construction Kit, 1958 model. I understand that it is guaranteed by you and may be returned in seven days for a full refund if I am not satisfied.

☐ I have enclosed $19.95 (plus 80¢ shipping in U. S., $1.50 west of Miss., $2.00 foreign). 3% New York City Sales Tax for N. Y. C. Residents.

☐ Send GENIAC C.O.D. I will pay postman the extra C.O.D. charge.



OVER 30,000 SOLD

We are proud to announce that over 30,000 **GENIACS** are in use by satisfied customers—schools, colleges, industrial firms and private individuals—a tribute to the skill and design work which makes it America's leading scientific kit. People like yourself with a desire to inform themselves about the computer field know that **GENIAC** is the only method for learning that includes both materials and texts and is devoted exclusively to the problems faced in computer study.

You are safe in joining this group because you are fully protected by our guarantee, and have a complete question and answer service available at no cost beyond that of the kit itself. You share in the experience of 30,000 kit users which contributes to the success of the 1958 **GENIAC**—with **DESIGN-O-Mat®** the exclusive **product of Oliver Garfield, Co., Inc.**, a Geniac is truly the most complete and unique kit of its kind in the world.

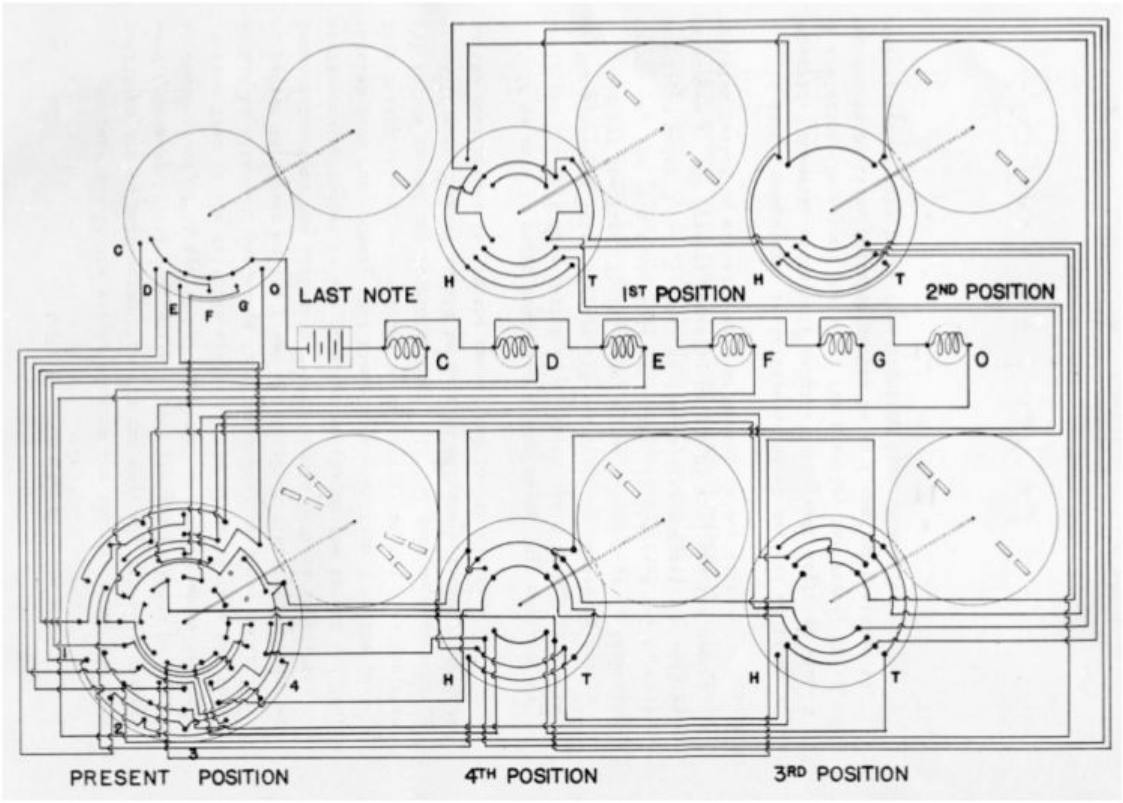
COMMENTS BY CUSTOMERS

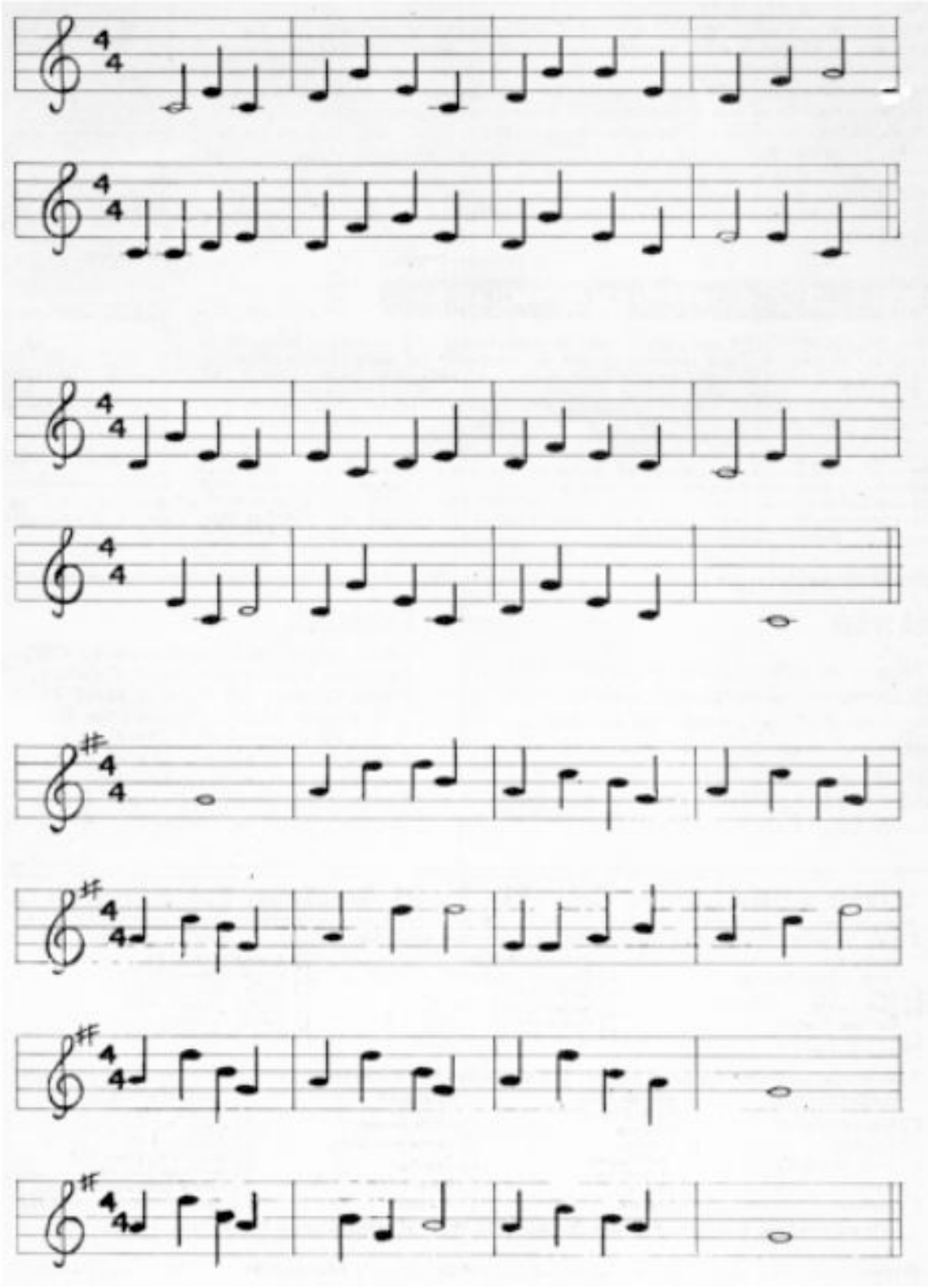
"Several months ago I purchased your **GENIAC** *Kit and found it an excellent piece of equipment. I learned a lot about computers from the enclosed books and pamphlets and I am now designing a small relay computer which will include arithmetical and logical units . . . another of my pet projects in cybernetics is a weather forecaster. I find that your* **GENIAC** *Kit may be used in their construction. I enclose the circuits and their explanation."* Eugene Darling, Malden

The **1958 GENIAC** comes with books and manuals and over 400 components.

1) A 64-page book "Simple Electric Brains and How to Make Them."
2) Beginners Manual—which outlines for people with no previous experience how to create electric circuits.
3) "A Symbolic Analysis of Relay and Switching Circuits."
4) DESIGN-O-Mat® over 50 new circuits outlines the practical principles of circuit design.
5. GENIAC STUDY GUIDE a complete course in computer fundamentals; guides the user to more advanced literature.

Plus all the components necessary for the building of over 125 machines and as many others as you can design yourself.





Courtesy of John F. Sowa. Used with permission.

- 1961: Harry Olson and Herbert Belar build a sophisticated electronic machine that produced and synthesized melodices based on Markovian pitch and rhythm analysis of eleven Stephen Collins Foster songs (1961)

TABLE II. Two-note sequences of eleven Stephen Foster songs. Probability of note following the preceding note expressed in sixteenths.

	Probability of following note											
Note	b	c♯	d	e	F♯	G	G♯	A	B	C♯	D	E
b			16									
c♯			16									
d	1	1	2	5	3	1		1		1	1	
e		1	6	3	4			1			1	
F♯			2	4	5	2		2	1			
G					4	3		6	3			
G♯								16				
A			1		5	1	1	4	3		1	
B			1		1	1		9	2		2	
C♯									8		8	
D								4	7	3	1	1
E								6		10		

Source: Olson, H. F., and H. Belar. "Aid to Music Composition Employing a Random Probability System." *J. Acoust. Soc. Am.* 33, no. 9 (1961): 1163-1170.

TABLE III. Three-note sequences of eleven Stephen Foster songs. Probability of note following a dinote expressed in sixteenths.

Dinote	b	c♯	d	e	F♯	G	G♯	A	B	C♯	D	E
bd			16									
c♯d			5	6				5				
db			16									
dc♯			16									
dd		2	2	9	2	1						
de			3	4	8			1				
dF♯				7	3	2		4				
dG					11				5			
dA					4			12				
dc♯											16	
dD								2	11	3		
ec♯			16									
ed	1		1	4	5			1		1	3	
ee		1	12	1	2							
eF♯			1	3	6	4		1	1			
eA								13	3			
eD										16		
F♯d				12	3	1						
F♯e		2	7	3	2			1			1	
F♯F♯			3	4	6	2		1				
F♯G					4	3		6	3			
F♯A					2			10	3		1	
F♯B								16				
GF♯				8		8						
GG						8		8				
GA			2					10			4	
GB								16				
G♯A									16			
Ad				11	5							
AF♯			5	4	3	1		2	1			
AG					16							
AG♯								16				
AA					4	1	1	5	5			
AB			1		1			12	1		1	
AD								6	5	3	2	
Bd			16									
BF♯				11	5							
BG									16			
BA			1		9	1		2	1		2	
BB					2			12			2	
BD								9	2	5		
C♯B								16				
C♯D									6			10
DA					14			2				
DB						1		5	6		4	
DC♯									12		4	
DD									16			
DE								5		11		
EA								16				
EC♯											16	

Source: Olson, H. F., and H. Belar. "Aid to Music Composition Employing a Random Probability System." *J. Acoust. Soc. Am.* 33, no. 9 (1961): 1163-1170.

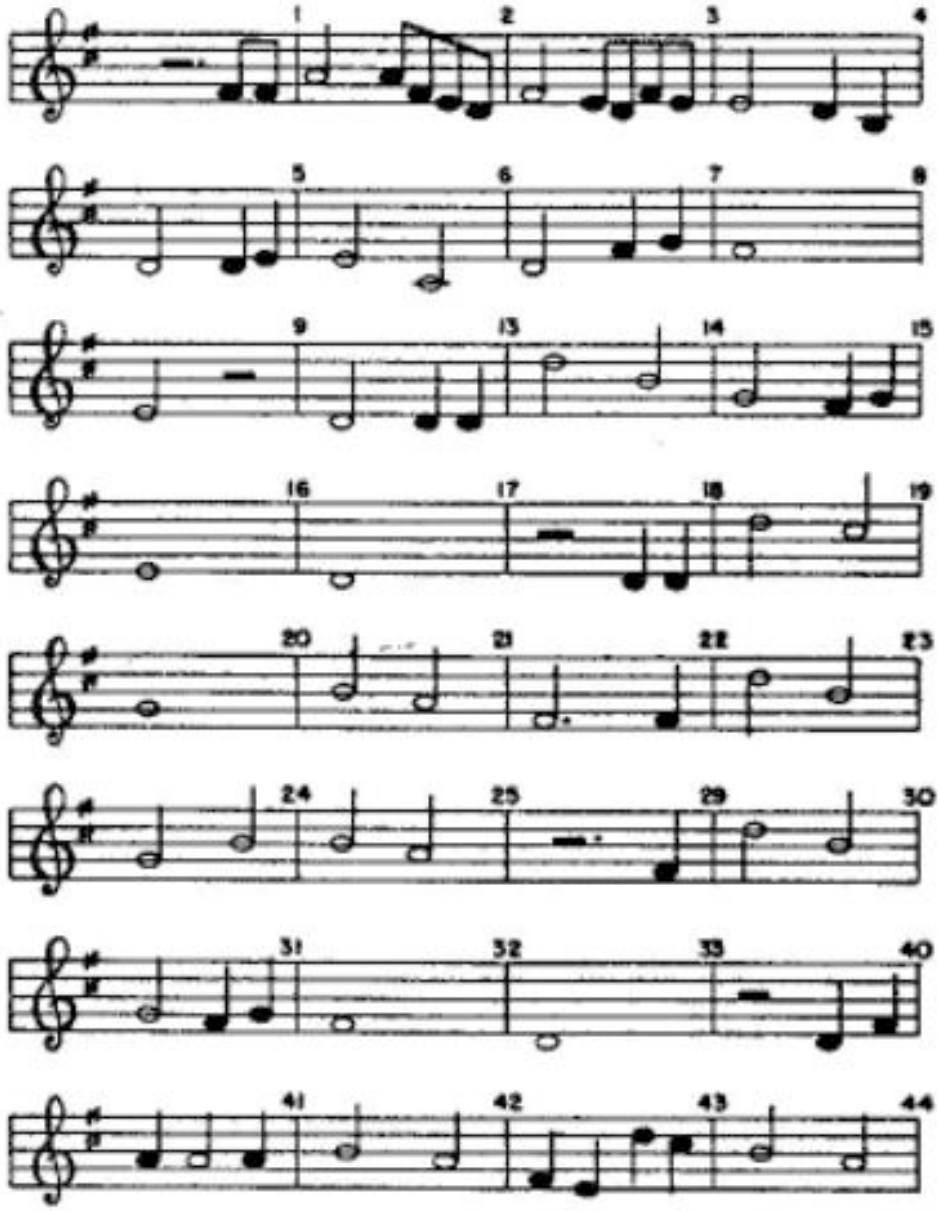
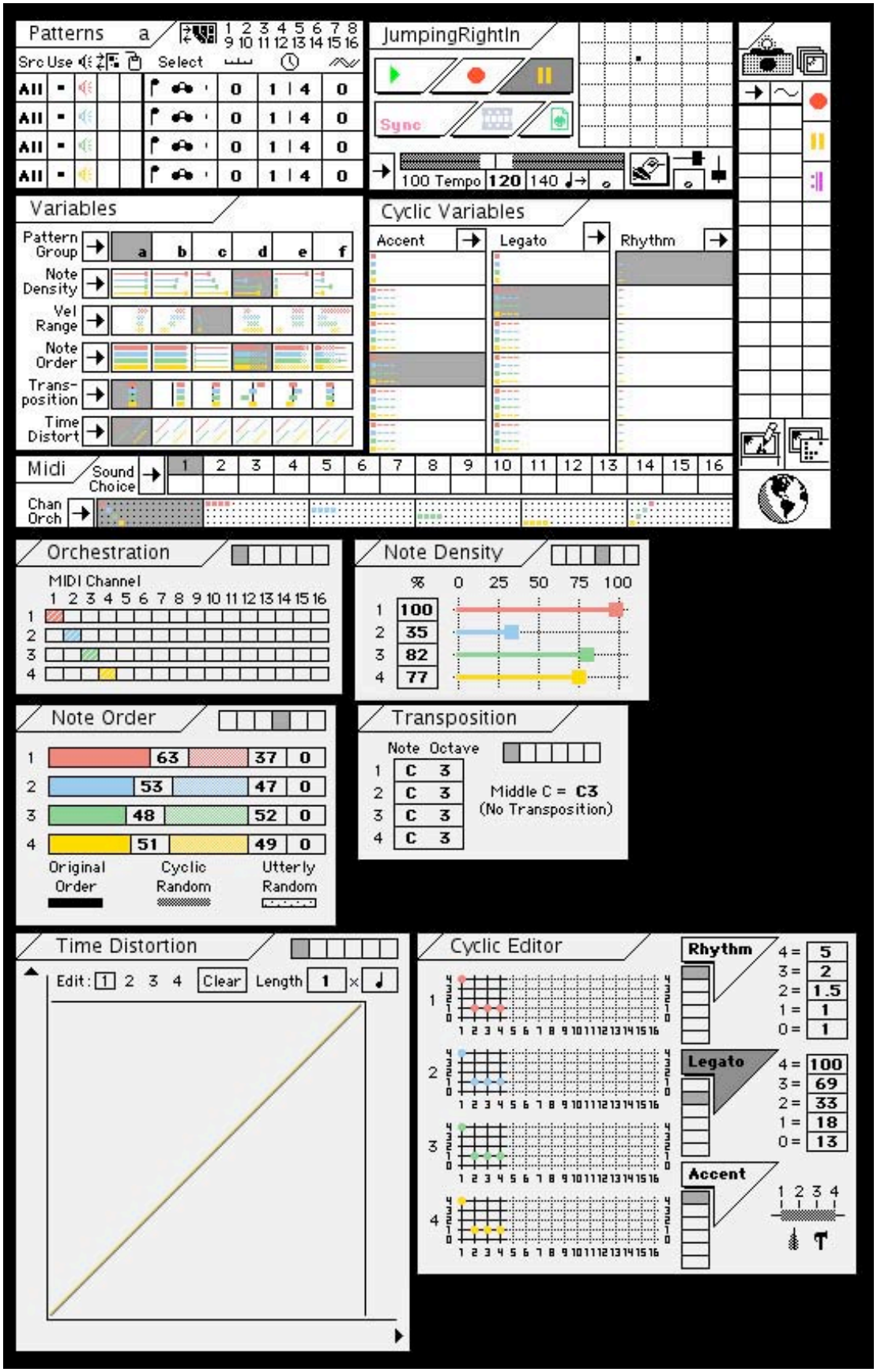


FIG. 11. Selected phrases from the output of the music composing machine. Set-up as of June 30, 1951, 4/4 time. Trinote probability derived from 11 Stephen Foster songs. Note, out of a total of 44 measures from the machine the following were selected, namely, 1 to 9, 13 to 25, 29 to 33, and 40 to 44 inclusive, and the following were ruled out, namely; 10 to 12, 26 to 28 and 34 to 39 inclusive.

Source: Olson, H. F., and H. Belar. "Aid to Music Composition Employing a Random Probability System." *J. Acoust. Soc. Am.* 33, no. 9 (1961): 1163-1170.

- David Zicarelli's Jam Factory and Joel Chadabe and Zicarelli's M (1987)



Courtesy of Cycling '74. Used with permission.

10.8. Markov Chains: Example: Shakespear

- Hamlet Act 3, Scene 1, Soliloquy

  YouTube (http://www.youtube.com/watch?v=-JD6gOrARk4)

- Shakespear: Hamlet: "To be or not to be"

```
To be, or not to be- that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune
Or to take arms against a sea of troubles,
And by opposing end them. To die- to sleep-
No more; and by a sleep to say we end
The heartache, and the thousand natural shocks
That flesh is heir to. 'Tis a consummation
Devoutly to be wish'd. To die- to sleep.
To sleep- perchance to dream: ay, there's the rub!
For in that sleep of death what dreams may come
When we have shuffled off this mortal coil,
Must give us pause. There's the respect
That makes calamity of so long life.
For who would bear the whips and scorns of time,
Th' oppressor's wrong, the proud man's contumely,
The pangs of despis'd love, the law's delay,
The insolence of office, and the spurns
That patient merit of th' unworthy takes,
When he himself might his quietus make
With a bare bodkin? Who would these fardels bear,
To grunt and sweat under a weary life,
But that the dread of something after death-
The undiscover'd country, from whose bourn
No traveller returns- puzzles the will,
And makes us rather bear those ills we have
Than fly to others that we know not of?
Thus conscience does make cowards of us all,
And thus the native hue of resolution
Is sicklied o'er with the pale cast of thought,
And enterprises of great pith and moment
With this regard their currents turn awry
And lose the name of action.- Soft you now!
The fair Ophelia!- Nymph, in thy orisons
Be all my sins rememb'red.
```

- 0-order Markov re-generation

  wish'd. contumely, Be contumely, the Devoutly thus pangs by thy of fardels makes name consummation pale Who we to respect coil, the to be and To Nymph, Th' That No 'Tis There's And the cowards of that When the weary or To a against wrong, And name With th' we thought, the sins That To my wrong, off perchance those Be scorns To his a that With others The quietus currents fly wrong, weary that To traveller time, When have scorns wrong, pale traveller against of make scorns quietus of delay, sleep. awry With to currents in and With cast coil, But have may arms Th' take arrows and The those their to regard the end we coil, fortune take

- 1-order Markov re-generation

die to others that is the name of so long life. For in the will, And enterprises of great pith and scorns of thought, And enterprises of thought, And lose the proud man's contumely, The undiscover'd country, from whose bourn No more; and the whips and moment With this regard their currents turn awry And lose the proud man's contumely, The slings and sweat under a sleep perchance to dream: ay, there's the whips and scorns of office, and arrows of great pith and scorns of something after death what dreams may come When he himself might his quietus make With this regard their currents turn awry And makes us pause. There's the law's delay, The heartache, and arrows of

- 2-order Markov re-generation

  To be, or not to be wish'd. To die to sleep No more; and by a sleep to say we end The heartache, and the thousand natural shocks That flesh is heir to. 'Tis a consummation Devoutly to be wish'd. To die to sleep No more; and by a sleep to say we end The heartache, and the thousand natural shocks That flesh is heir to. 'Tis a consummation Devoutly to be wish'd. To die to sleep No more; and by a sleep to say we end The heartache, and the thousand natural shocks That flesh is heir to. 'Tis a consummation Devoutly to be that is the question: Whether 'tis nobler in the mind to suffer The slings and

- 3-order Markov re-generation

  the name of action. Soft you now! The fair Ophelia! Nymph, in thy orisons Be all my sins rememb'red. To be, or not to be wish'd. To die to sleep No more; and by a sleep to say we end The heartache, and the thousand natural shocks That flesh is heir to. 'Tis a consummation Devoutly to be that is the question: Whether 'tis nobler in the mind to suffer The slings and arrows of outrageous fortune Or to take arms against a sea of troubles, And by opposing end them. To die to sleep No more; and by a sleep to say we end The heartache, and the spurns That patient merit of th' unworthy takes, When he himself

- 4-order Markov re-generation

  those ills we have Than fly to others that we know not of? Thus conscience does make cowards of us all, And thus the native hue of resolution Is sicklied o'er with the pale cast of thought, And enterprises of great pith and moment With this regard their currents turn awry And lose the name of action. Soft you now! The fair Ophelia! Nymph, in thy orisons Be all my sins rememb'red. To be, or not to be wish'd. To die to sleep No more; and by a sleep to say we end The heartache, and the thousand natural shocks That flesh is heir to. 'Tis a consummation Devoutly to be that is the question: Whether 'tis nobler in the

- 5-order Markov re-generation

  we have shuffled off this mortal coil, Must give us pause. There's the respect That makes calamity of so long life. For who would bear the whips and scorns of time, Th' oppressor's wrong, the proud man's contumely, The pangs of despis'd love, the law's delay, The insolence of office, and the spurns That patient merit of th' unworthy takes, When he himself might his quietus make With a bare bodkin? Who would these fardels bear, To grunt and sweat under a weary life, But that the dread of something after death The undiscover'd country, from whose bourn No

traveller returns puzzles the will, And makes us rather bear those ills we have Than fly to others that we know

10.9. Markov Chains: Example: Mozart Symphony 40

- Audio: Mozart: Symphony 40
- Pitch and rhythm based Markov regeneration at various orders
- Markov-generated examples [markovMozart.py]

10.10. Markov Analysis and Generation with athenaCL Python Libraries: Text

- Use the athenaCL Markov module
- Create a markov.Transition instances to do analysis
- Example: string data [markovShakespear.py]

```
import random
from athenaCL.libATH import markov

src =  """To be, or not to be- that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune
Or to take arms against a sea of troubles,
And by opposing end them. To die- to sleep-
No more; and by a sleep to say we end
The heartache, and the thousand natural shocks
That flesh is heir to. 'Tis a consummation
Devoutly to be wish'd. To die- to sleep.
To sleep- perchance to dream: ay, there's the rub!
For in that sleep of death what dreams may come
When we have shuffled off this mortal coil,
Must give us pause. There's the respect
That makes calamity of so long life.
For who would bear the whips and scorns of time,
Th' oppressor's wrong, the proud man's contumely,
The pangs of despis'd love, the law's delay,
The insolence of office, and the spurns
That patient merit of th' unworthy takes,
When he himself might his quietus make
With a bare bodkin? Who would these fardels bear,
To grunt and sweat under a weary life,
But that the dread of something after death-
The undiscover'd country, from whose bourn
No traveller returns- puzzles the will,
And makes us rather bear those ills we have
Than fly to others that we know not of?
```

```
Thus conscience does make cowards of us all,
And thus the native hue of resolution
Is sicklied o'er with the pale cast of thought,
And enterprises of great pith and moment
With this regard their currents turn awry
And lose the name of action.- Soft you now!
The fair Ophelia!- Nymph, in thy orisons
Be all my sins rememb'red."""

orderMax = 2 # large numbers here will take time!
mkObj = markov.Transition()
mkObj.loadString(src, orderMax) # source and max order1

for order in range(0, orderMax+1):
   print('requested order: ' + order)
   msg = []
   for x in range(120):
      val = random.random()
      msg.append(mkObj.next(val, msg, order))
   print(' '.join(msg) + '\n')
```

10.11. Markov Analysis and Generation with athenaCL Python Libraries: MIDI

- Example: pitch and rhythm data [markovMozart.py]

```
import os, random, sys
from athenaCL.libATH import midiTools
from athenaCL.libATH import osTools
from athenaCL.libATH import pitchTools
from athenaCL.libATH import rhythm
from athenaCL.libATH import markov
from athenaCL.libATH.libOrc import generalMidi
from athenaCL.libATH.libPmtr import parameter
from athenaCL.libATH.libPmtr import basePmtr

OUTDIR = '/Volumes/xdisc/_scratch'
BEATDUR = rhythm.bpmToBeatTime(128) # provide bpm value

def getInstName(nameMatch):
   for name, pgm in generalMidi.gmProgramNames.items():
      if name.lower().startswith(nameMatch.lower()):
         return pgm # an integer
   return None

def convertPitch(src, octShift):
   post = []
   for pitch in src:
      midiPs = pitchTools.psToMidi(pitchTools.psNameToPs(pitch))
      midiPs = midiPs + (12*octShift)
      post.append(midiPs)
   return post # a list of integers

def convertRhythm(src, scale):
   post = []
   for rhythm in src:
      post.append(rhythm*scale)
   return post # a list of integers

def mozartMarkov(events, order, octaveShift, rhythmScale):
   pitchSequence = [
```

```
        'E$5','D5','D5','E$5','D5','D5','E$5','D5','D5',
        'B$5','B$5','A5','G5','G5','F5','E$5','E$5','D5','C5','C5',
        'D5','C5','C5', 'D5','C5','C5', 'D5','C5','C5',
        'A5','A5','G5','G$5','G$5','E$5','D5','D5','C5','B$4','B$4',
        'B$5','A5','A5','C6','G$5','A5','G5','D5',
        'B$5','A5','A5','C6','G$5','A5','G5','B$5','A5','G5','F5','E$5',
        'D5','D$5','D5',
        'D4','D4','D4', 'D4','D4','D4',
        'D4','D4','D4', 'D4','D4','D4', 'D4','D4','D4']

    rhtyhmSequence = [
        .5, .5, 1, .5, .5, 1, .5, .5, 1, 1,
        .5, .5, 1, .5, .5, 1, .5, .5, 1, 2,
        .5, .5, 1, .5, .5, 1, .5, .5, 1,
        2, .5, .5, 1, .5, .5, 1, .5, .5, 1, 2,
        .5, .5, 1, 1, 1, 1, 1, 2,
        .5, .5, 1, 1, 1, 1, 1, 1, .5, .5, .5, .5,
        4, 4, 3,
        .5, .5, 3, .5, .5, 3,
        .5, .5, 1, .5, .5, 1, .5, .5, 1]

    mkPitch = markov.Transition()
    mkRhythm = markov.Transition()
    mkPitch.loadList(convertPitch(pitchSequence, octaveShift), order)
    mkRhythm.loadList(convertRhythm(rhtyhmSequence, rhythmScale), order)

    pitchHistory = []
    rhythmHistory = []

    ampGen = parameter.factory(['ws','e',4,0,100,120]) # sine osc b/n 90 and 120
    f = random.choice(range(50,70))
    phase = random.random()
    panGen = parameter.factory(['ws','e',f,phase,20,107])
    score = []
    tStart = 0.0

    for i in range(events):
        pitch = mkPitch.next(random.random(), pitchHistory, order)
        pitchHistory.append(pitch)
        rhythm = mkRhythm.next(random.random(), rhythmHistory, order)
        rhythmHistory.append(rhythm)

        dur = BEATDUR * rhythm
        amp = int(round(ampGen(0)))
        pan = int(round(panGen(0)))
        event = [tStart, dur, amp, pitch, pan]
        score.append(event)
        tStart += dur
    return score

def main(order):
    trackList = []
    score = mozartMarkov(100, order, -1, 1)
    trackList.append(['part-a',  getInstName('piano'), None, score])
    path = os.path.join(OUTDIR, 'test.midi')
    mObj = midiTools.MidiScore(trackList)
    mObj.write(path) # writes in cwd
    osTools.openMedia(path)

if __name__ == '__main__':
    if len(sys.argv) != 2:
        print("args: order")
    else:
        main(int(sys.argv[1]))
```

10.12. Reading: Ariza: Beyond the Transition Matrix: A Language-Independent, String-Based Input Notation for Incomplete, Multiple-Order, Static Markov Transition Values

- Ariza, C. 2006. "Beyond the Transition Matrix: A Language-Independent, String-Based Input Notation for Incomplete, Multiple-Order, Static Markov Transition Values." Internet: http://www.flexatone.net/docs/btmimosmtv.pdf.

- What are some potential advantages of the transition string over the transition matrix?

- Why might modulating Markov order be desirable?

10.13. Utility Markov Analysis and Generation within athenaCL

- AUma command can be used to get an analysis string for an space-separated sequence

```
:: auma
maximum analysis order: 1
enter space-separated string: 0 1 1 1 1 0 1 2 3 4 0 0 2 1 3 2 4 0 0
AthenaUtility Markov Analysis
a{0}b{1}c{2}d{3}e{4}:{a=6|b=6|c=3|d=2|e=2}a:{a=3|b=2|c=1}b:{a=1|b=3|c=1|d=1}c:{b=1|d
=1|e=1}d:{c=1|e=1}e:{a=2}
```

- AUmg command can be used to use a transition string to generate values

```
:: aumg
number of generations: 20
desired order: 1
enter Markov transition string:
a{0}b{1}c{2}d{3}e{4}:{a=6|b=6|c=3|d=2|e=2}a:{a=3|b=2|c=1}b:{a=1|b=3|c=1|d=1}c:{b=1|d
=1|e=1}d:{c=1|e=1}e:{a=2}
AthenaUtility Markov Generator
4,0,1,1,1,1,1,3,2,1,1,1,1,1,2,4,0,0,1,0
```

10.14. Markov-Based Proportional Rhythm Generation

- The MarkovPulse Generator permits specifying proportional rhythms (pulse truples) as Markov states

```
:: tpv markovpulse
Rhythm Generator ParameterObject
{name,documentation}
MarkovPulse         markovPulse, transitionString, parameterObject
                    Description: Produces Pulse sequences by means of a Markov
                    transition string specification and a dynamic transition
                    order generator. The Markov transition string must define
                    symbols that specify valid Pulses. Markov transition order
                    is specified by a ParameterObject that produces values
                    between 0 and the maximum order available in the Markov
                    transition string. If generated-orders are greater than
                    those available, the largest available transition order will
                    be used. Floating-point order values are treated as
                    probabilistic weightings: for example, a transition of 1.5
```

```
                    offers equal probability of first or second order selection.
                    Arguments: (1) name, (2) transitionString, (3)
                    parameterObject {order value}
```

- Command sequence:
  - emo mp
  - tin a 64
  - *simple zero-order selection*

    tie r mp,a{4,1}b{4,3}c{4,5}d{4,7}:{a=4|b=3|c=2|d=1}
  - *first order generation that encourages movement toward the shortest duration*

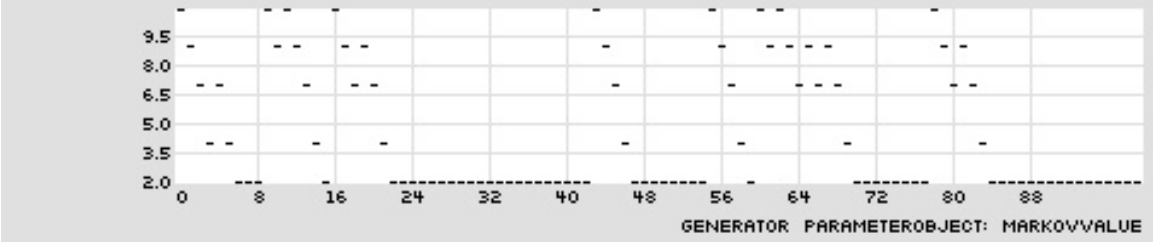
    tie r mp,a{8,1}b{4,3}c{4,7}d{4,13}a:{a=9|d=1}b:{a=5|c=1}c:{b=1}d:{c=1},(c,1)
  - eln; elh

10.15. Markov-Based Value Generation

- The MarkovValue Generator permits specifying any value as Markov states, and dynamically moving between different Markov orders

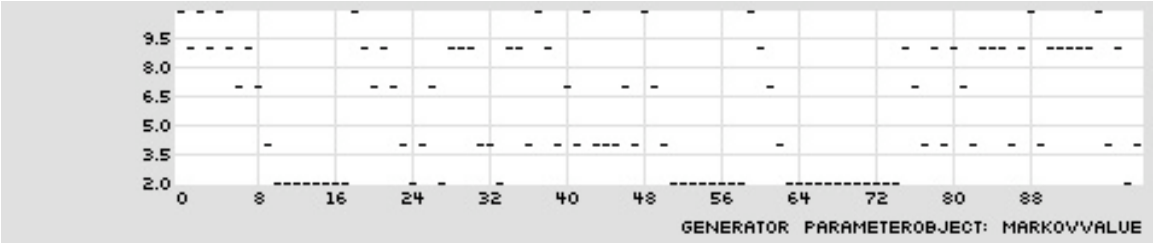
```
:: tpv mv
Generator ParameterObject
{name,documentation}
MarkovValue         markovValue, transitionString, parameterObject
                    Description: Produces values by means of a Markov transition
                    string specification and a dynamic transition order
                    generator. Markov transition order is specified by a
                    ParameterObject that produces values between 0 and the
                    maximum order available in the Markov transition string. If
                    generated-orders are greater than those available, the
                    largest available transition order will be used. Floating-
                    point order values are treated as probabilistic weightings:
                    for example, a transition of 1.5 offers equal probability of
                    first or second order selection. Arguments: (1) name, (2)
                    transitionString, (3) parameterObject {order value}

:: tpmap 100
mv,a{2}b{4}c{7}d{9}e{11}:{a=1|b=3|c=1|d=3|e=1}a:{a=9|e=1}b:{a=3|c=1}c:{b=3|d=1}d:{c=
3|e=1}e:{d=1},(c,1)
markovValue, a{2}b{4}c{7}d{9}e{11}:{a=1|b=3|c=1|d=3|e=1}a:{a=9|e=1}b:{a=3|c=1}c:
{b=3|d=1}d:{c=3|e=1}e:{d=1}, (constant, 1)
TPmap display complete.
```



- The modulating the order of the Markov chain can create dynamic long-range behavior

```
:: tpmap 100
mv,a{2}b{4}c{7}d{9}e{11}:{a=1|b=3|c=1|d=3|e=1}a:{a=9|e=1}b:{a=3|c=1}c:{b=3|d=1}d:{c=
3|e=1}e:{d=1},(wp,e,50,0,1,0)
markovValue, a{2}b{4}c{7}d{9}e{11}:{a=1|b=3|c=1|d=3|e=1}a:{a=9|e=1}b:{a=3|c=1}c:
{b=3|d=1}d:{c=3|e=1}e:{d=1},
TPmap display complete.
```



- Command sequence:
  - emo m
  - tin a 26
  - *rhythm generated with absolute values via ConvertSecond and a dynamic WaveHalfPeriodSine generator*

    tie r cs,(whps,e,(bg,rp,(5,10,15,20)),0,.200,.050)
  - *first-order selection*

    tie f
    mv,a{2}b{4}c{7}d{9}e{11}:{a=1|b=3|c=1|d=3|e=1}a:{a=9|e=1}b:{a=3|c=1}c:{b=3|d=1}d:{c=3|e=1}e:{d=1},(c,1)
  - *dynamic first and zero order selection*

    tie f
    mv,a{2}b{4}c{7}d{9}e{11}:{a=1|b=3|c=1|d=3|e=1}a:{a=9|e=1}b:{a=3|c=1}c:{b=3|d=1}d:{c=3|e=1}e:{d=1},(wp,e,100,0,1,0)
  - *zero-order Markov amplitude values*

tie a mv,a{.4}b{.6}c{.8}d{1}:{a=6|b=4|c=3|d=1}

- *amplitude values scaled by a dynamic WaveHalfPeriodPulse*

  tie a om,(mv,a{.4}b{.6}c{.8}d{1}:{a=6|b=4|c=3|d=1}),(whpp,e,(bg,rp,(5,15,10)))

- *octave values are provided by a first-order Markov chain*

  tie o mv,a{0}b{-1}c{-2}d{-3}a:{a=9|d=1}b:{a=3|b=1}c:{b=3|c=1}d:{c=1},(c,1)

- tie t 0,60

- eln; elh

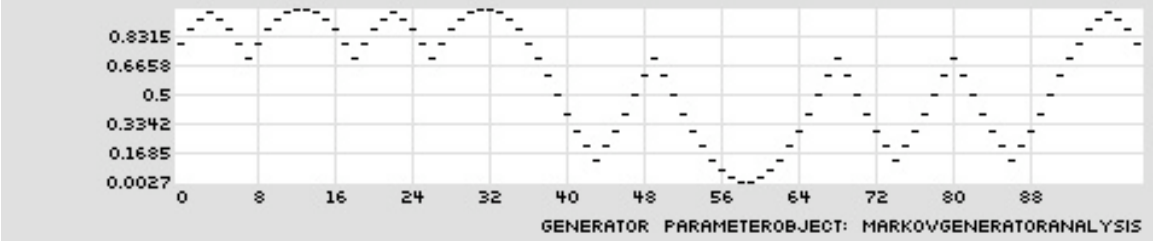
10.16. Markov-Based Combined Analysis and Generation

- The MarkovGeneratorAnalysis Generator permits using the output of a ParameterObject as the source for Markov analysis

```
:: tpv mga
Generator ParameterObject
{name,documentation}
MarkovGeneratorAnalysis markovGeneratorAnalysis, parameterObject, valueCount,
                        maxAnalysisOrder, parameterObject
                        Description: Produces values by means of a Markov
                        analysis of values provided by a source Generator
                        ParameterObject; the analysis of these values is used
                        with a dynamic transition order Generator to produce new
                        values. The number of values drawn from the source
                        Generator is specified with the valueCount argument. The
                        maximum order of analysis is specified with the
                        maxAnalysisOrder argument. Markov transition order is
                        specified by a ParameterObject that produces values
                        between 0 and the maximum order available in the Markov
                        transition string. If generated-orders are greater than
                        those available, the largest available transition order
                        will be used. Floating-point order values are treated as
                        probabilistic weightings: for example, a transition of
                        1.5 offers equal probability of first or second order
                        selection. Arguments: (1) name, (2) parameterObject
                        {source Generator}, (3) valueCount, (4)
                        maxAnalysisOrder, (5) parameterObject {output order
                        value}
```

- First order analysis and regeneration of a sine oscillation

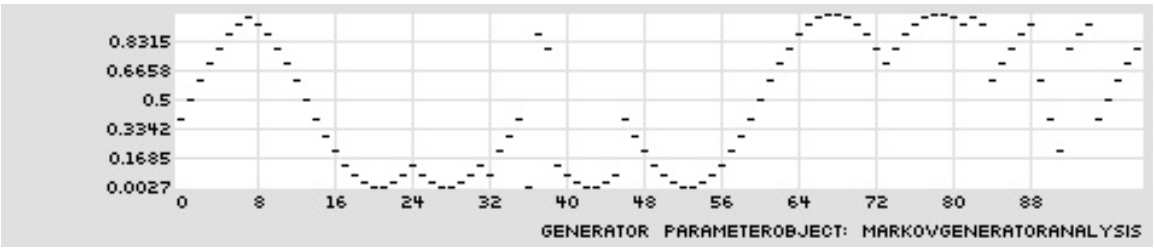
```
:: tpmap 100 mga,(ws,e,30),30,2,(c,1)
markovGeneratorAnalysis, (waveSine, event, (constant, 30), 0, (constant, 0),
(constant, 1)), 30, 2, (constant, 1)
TPmap display complete.
```



- Analysis and regeneration of a sine oscillation with dynamic orders from 0.5 to 1.5

  Floating-point orders are treated as probabilistic weightings toward nearest integers

```
:: tpmap 100 mga,(ws,e,30),30,2,(ws,e,50,0,0.5,1.5)
markovGeneratorAnalysis, (waveSine, event, (constant, 30), 0, (constant, 0),
(constant, 1)), 30, 2, (waveSine, event, (constant, 50), 0, (constant, 0.5),
(constant, 1.5))
TPmap display complete.
```



10.17. Resuming PD Tutorial

- PD Tutorial

Chapter 11. Meeting 11, Workshop

11.1. Announcements

- Musical Design Report 2 due today, 11 March
- Quiz next Tuesday
- Next week, and after spring break: make appointments with me to talk about sonic system projects.

11.2. Workshop: Musical Design Report 2

- `Three students presenting today`

11.3. Configuring Event Outputs

- For each EventMode, some output is always created (in EventMode midiPercussion, a midi file is always created)
- Some outputs are independent of any EventMode (an xmlAthenaObject can be always generated)
- A list of desired outputs (EventOutputs) is always consulted to determine what required and what optional outputs are created when creating an EventList
- The EOls command can be used to see what EventOutputs are active

```
:: eols
EventOutput active:
{name}
   acToolbox
   audioFile
   csoundBatch
   csoundData
   csoundOrchestra
   csoundScore
   midiFile
   pureDataArray
   superColliderTask
   textSpace
   textTab
   xmlAthenaObject
```

- When working with Csound, it is always desirable to have csoundData selected, as this causes the creation of integrated CSD files (combined orchestra and score files). Use the EOo to select active EventOutputs.

```
:: eoo cd
EventOutput formats: csoundData.

:: eols
EventOutput active:
{name}
   acToolbox
   audioFile
   csoundBatch
 + csoundData
   csoundOrchestra
   csoundScore
   midiFile
   pureDataArray
   superColliderTask
   textSpace
   textTab
   xmlAthenaObject
```

- It is desirable to write an AthenaObject XML file (xmlAthenaObject) when creating an EventList to premit reloading an athenaCL session.

```
:: eoo xao
EventOutput formats: csoundData, xmlAthenaObject.

:: eols
EventOutput active:
{name}
   acToolbox
   audioFile
   csoundBatch
 + csoundData
   csoundOrchestra
   csoundScore
   midiFile
   pureDataArray
   superColliderTask
   textSpace
   textTab
 + xmlAthenaObject
```

11.4. A Noise Instrument

- Csound instruments add auxiliary parameter fields to Textures
- Such parameter fields permit control of synthesis parameters
- Command sequence:
  - emo cn
  - tin a 13
  - tie r cs,(whps,e,(bg,rp,(5,10,15,20)),0,.200,.050)
  - *set initial low-pass filter cutoff frequency*

tie x2 whps,e,(bg,rp,(5,10,20,2,10)),0,400,18000

- *set final low-pass filter cutoff frequency*

  tie x3 whps,e,(bg,rp,(5,10,20,2,10)),0,400,18000

- *panning controlled by fractional noise with infrequent zero-order Markov controlled jumps out of 1/f2 to 1/f0*

  tie n n,100,(mv,a{2}b{0}:{a=12|b=1}),0,1

- eln; elh

11.5. A Sample Playback Instrument

- Csound instruments add auxiliary parameter fields to Textures
- Such parameter fields permit control of synthesis parameters
- Command sequence:
  - emo cn
  - tin a 32
  - *set a file path to an audio file*

    tie x6 cf,/Volumes/xdisc/_sync/_x/src/martingale/martingale/audio/29561.aif

  - *line segment absolute rhythm durations*

    tie r cs,(ls,e,(ru,5,30),(ru,.03,.15),(ru,.03,.15))

  - *start position within audio file in seconds*

    tie x5 ru,0,40

  - tie a ls,e,(bg,rc,(3,5,20)),.1,1
  - tie x2 whps,e,(bg,rp,(5,10,20,2,10)),0,100,10000
  - eln; elh

11.6. A Sample Playback Instrument with Variable Playback Rate

- Csound instruments add auxiliary parameter fields to Textures
- Such parameter fields permit control of synthesis parameters

- Command sequence:
  - emo cn
  - tin a 230
  - *set a file path to an audio file*

    tie x6 cf,/Volumes/xdisc/_sync/_x/src/martingale/martingale/audio/32673.aif
  - *line segment absolute rhythm durations*

    tie r cs,(ls,e,(ru,10,30),(ru,.05,.25),(ru,.05,.25))
  - *start position within audio file in seconds*

    tie x5 ru,0,10
  - *initial and final audio playback rate*

    tie x7 mv,a{1}b{.75}c{.5}d{.2}e{2}:{a=6|b=3|c=2|d=1|e=1}

    tie x8 mv,a{1}b{.75}c{.5}d{.2}e{2}:{a=6|b=3|c=2|d=1|e=1}
  - *panning controlled by fractional noise with infrequent zero-order Markov controlled jumps out of 1/f2 to 1/f0*

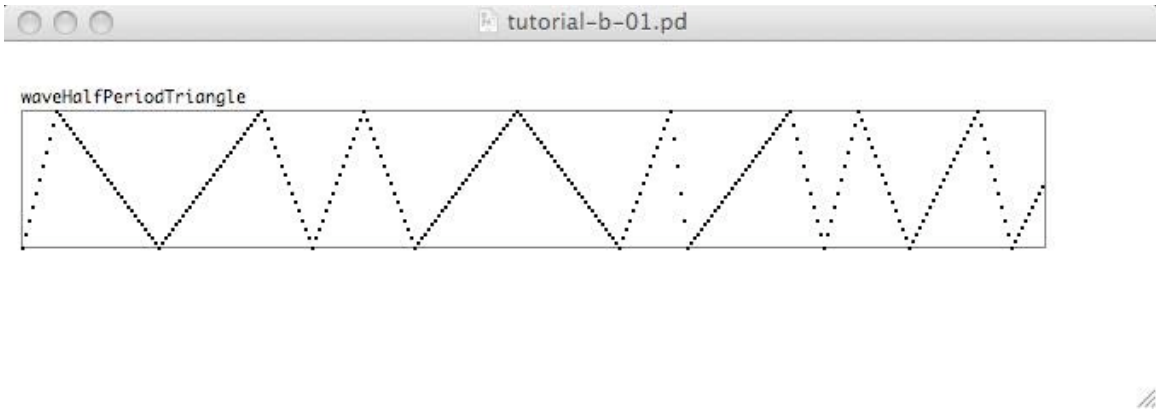
    tie n n,100,(mv,a{2}b{0}:{a=12|b=1}),0,1
  - *two instances simultaneously*

    ticp a b
  - eln; elh

11.7. Exporting ParameterObject output as PD Arrays

- ParameterObject outputs can be exported as PD Arrays.
- Can be used to produce control data used in PD processing
- The TPe (TextureParameter Export) command interactively

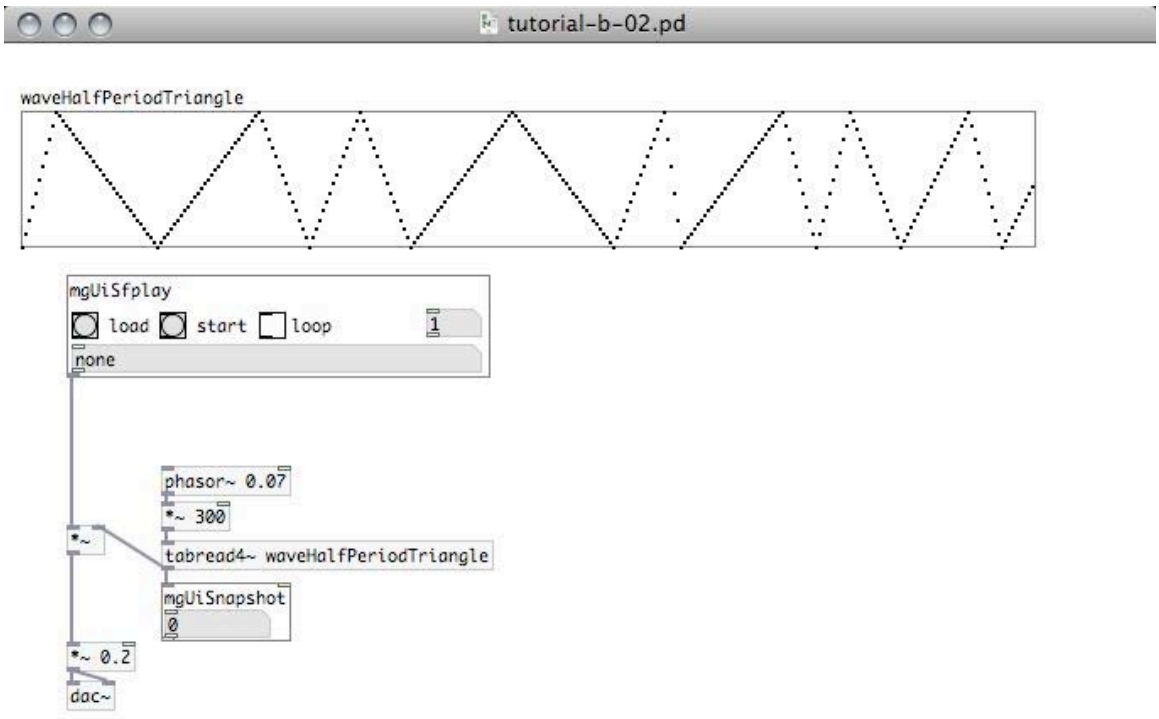
```
:: tpe
enter an export format: pda
number of events: 300
enter a Generator ParameterObject argument: whpt,e,(bg,rc,(5,10,15,20,30))
command.py: temporary file: /Volumes/xdisc/_scratch/ath2010.03.11.08.16.16.pd
complete: (/Volumes/xdisc/_scratch/ath2010.03.11.08.16.16.pd)
```



- Or as a single Command:
  - tpe pda 300 whpt,e,(bg,rc,(5,10,15,20,30))

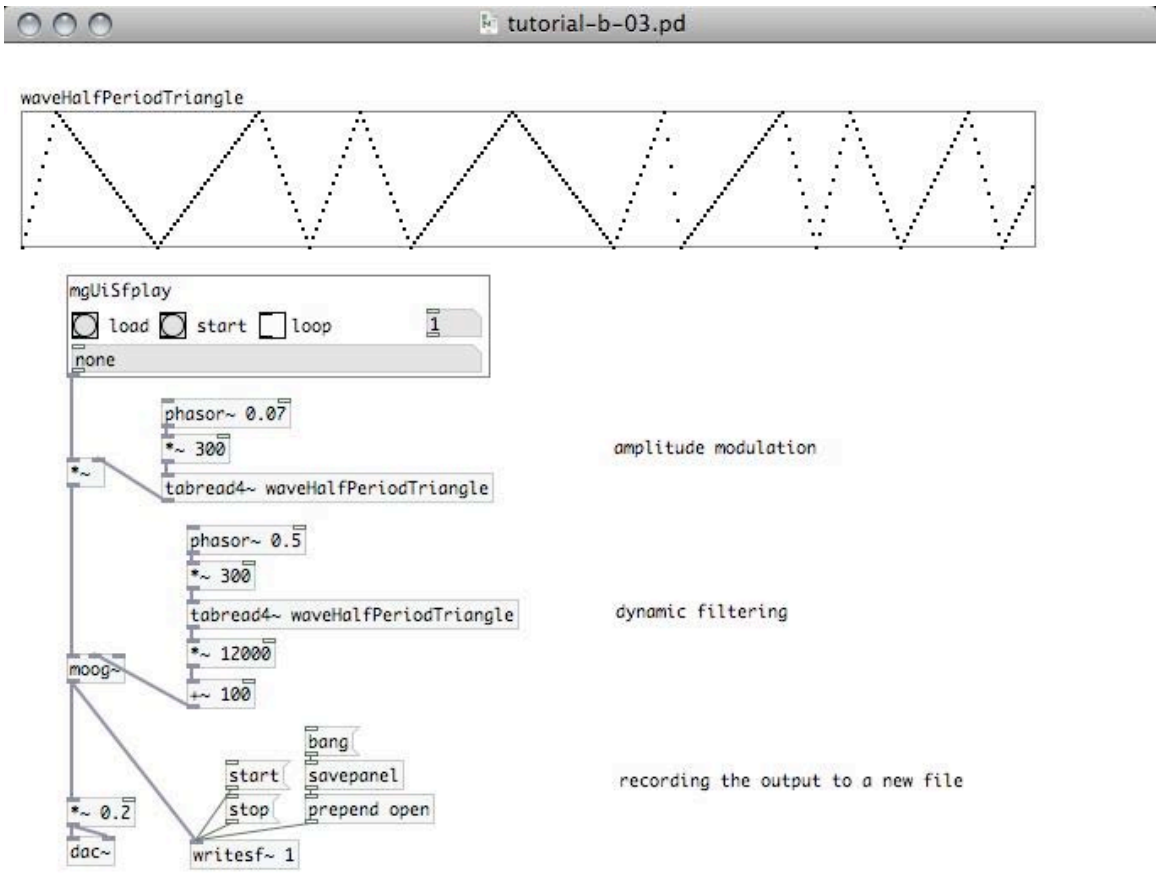
11.8. Using PD Arrays to Process Sound Files

- PD Arrays can be read at the audio rate by [tabread4~] objects
- The [tabread4~] object needs index values at the audio rate, best provided by a scaled [phasor~] (provide output cyclical output between 0 and 1)
- Scaling the amplitude by the PD Array



- Providing dynamic filtering with a [moog~] filter

  [writesf~] can be used to write a new audio file


tutorial-b-03.pd
waveHalfPeriodTriangle
mgUiSfplay
load
start
loop
1
none
phasor~ 0.07
*~ 300
*~
tabread4~ waveHalfPeriodTriangle
amplitude modulation
phasor~ 0.5
*~ 300
tabread4~ waveHalfPeriodTriangle
dynamic filtering
*~ 12000
moog~
+~ 100
bang
start
savepanel
recording the output to a new file
stop
prepend open
*~ 0.2
dac~
writesf~ 1

Chapter 12. Meeting 12, History: Iannis Xenakis

12.1. Announcements

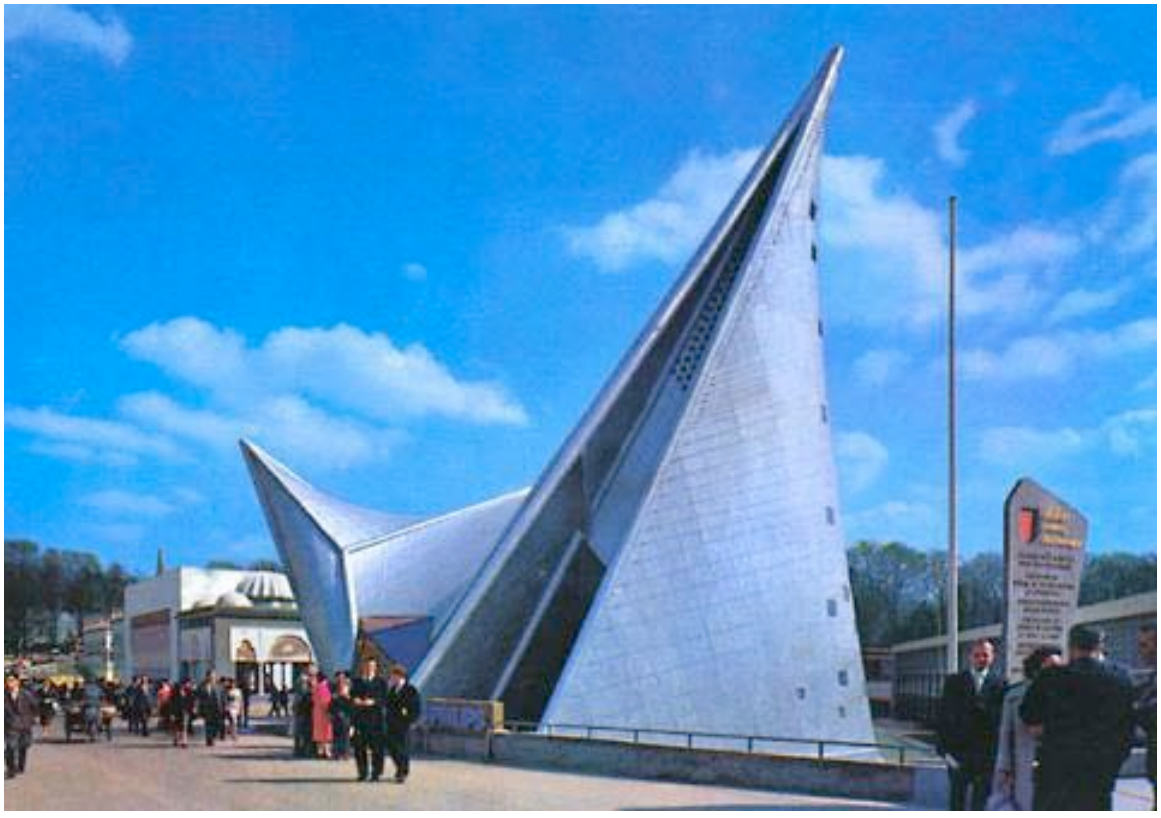
- Musical Design Report 3 due 6 April
- Start thinking about sonic system projects

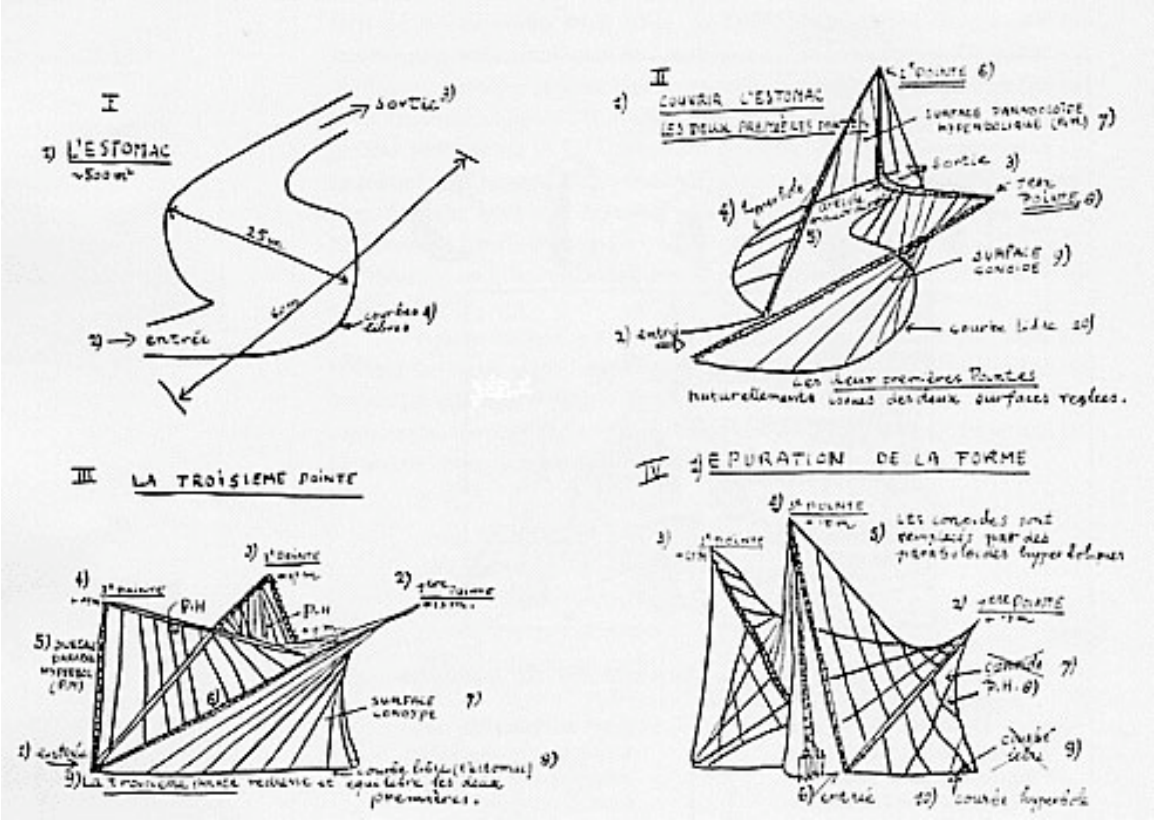
12.2. Quiz

- 10 Minutes

12.3. Xenakis: Background

- An architect, mathematician, music theorist, and composer
- 1958: Designed Philips Pavilion for Brussels Worlds Fair as assistant of Le Corbusier (1887-1965)





- Early tape music: Diamorphoses (1957), Concret PH (1958), Orient Occident (1960)
- Innovative early instrumental music based on geometries and procedures

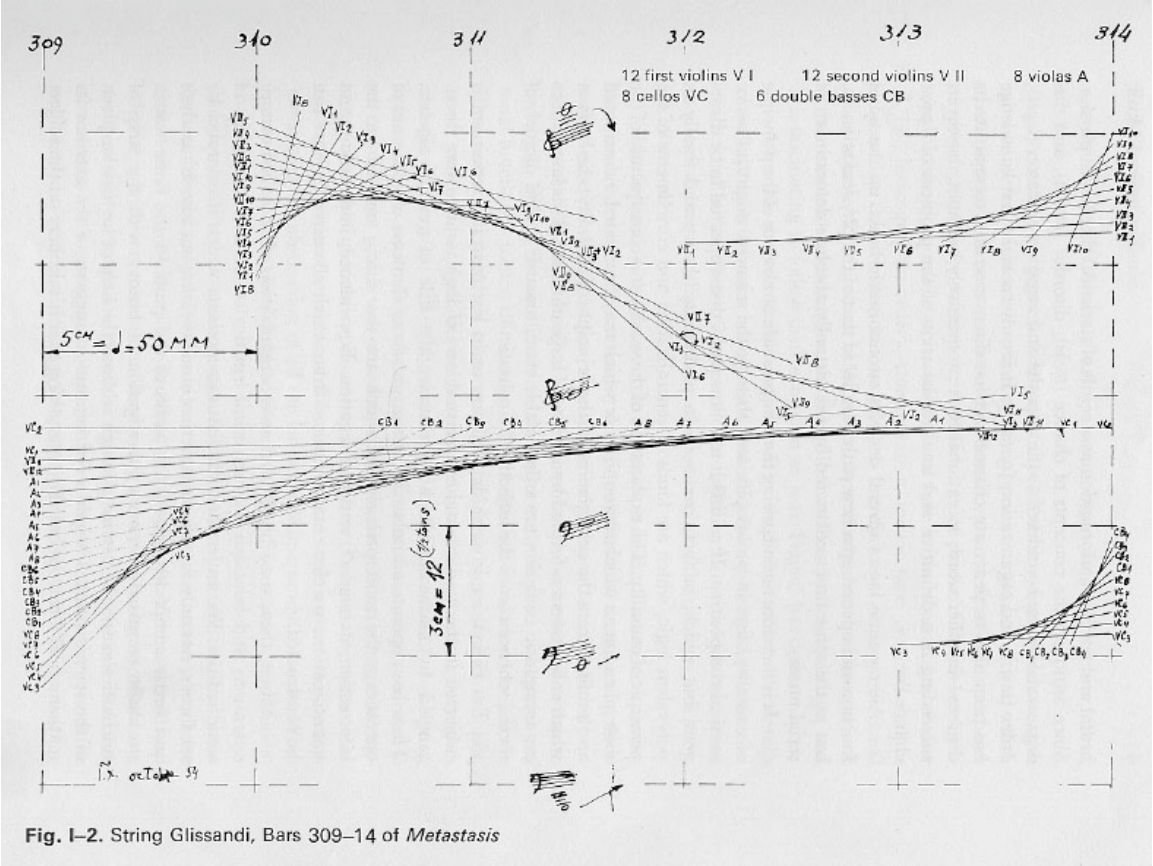


Fig. I–2. String Glissandi, Bars 309–14 of *Metastasis*

- Proposed models of granular synthesis after research of Gabor
- Proposed and developed a wide range of music technologies for creative applications

12.4. Xenakis: History

- Fought in Greek resistance to Nazi occupation during World War II
- Moved to France, began work with Le Corbusier, heard music of Schaeffer
- Studied composition with Olivier Messiaen
- 1955: "The Crisis of Serial Music" (Xenakis 1955)
- 1963: first edition of text *Formalized Music*
- 1967-1972: professor at Indiana University, Bloomington
- 1972: creates the Centre d'Etudes de Mathematiques et Automatiques Musicale (CEMAMu) near Paris
- 1972-1989: professor at Sorbonne University in Paris

12.5. Xenakis: Pithoprakta and Achorripsis

- *Pithoprakta* (1955-56) and *Achorripsis* (1956-57): composed with systems based on probability and statistics
- Employed techniques of "stochastic music": specify statistical trends, densities, and ranges rather then all note parameters
- A procedural approach to composition
- A response to the "Crisis of Serialism" (Xenakis 1955)
- "But other paths also led to the same stochastic crossroads -- first of all, natural events such as the collision of hail or rain with hard surfaces, or the song of cicadas in a summer field. These sonic events are made out of thousands of isolated sounds; this multitude of sounds, seen as a totality, is a new sonic event. This mass event is articulated and forms a plastic mold of time, which itself follows aleatory and stochastic laws." (Xenakis 1992, p. 9)

12.6. Listening: Xenakis

- Achorripsis, (1956-1957) [4:50 to 6:41]

12.7. Reading: Xenaxis, Xenakis on Xenakis

- Xenakis, I. 1987. "Xenakis on Xenakis." *Perspectives of New Music* 25(1-2): 16-63.
- What was Xenakis's early background in music and sound?
- Throughout his writings Xenakis talks about the pressures and problems of the Conservatory, Instruments, and Solfege: what is he referring to?
- Xenakis has particular relationship with the visual, graphical, and drawn approaches to thinking about music. Explain this relationship.
- In what ways does Xenakis imagine that technology will change the role of music in people's lives?

12.8. The Stochastic Music Program

- 1961: Xenakis gains access to an IBM 7090 at IBM France

Courtesy of IBM Corporate Archives. Used with permission.

- Programs the Stochastic Music Program (SMP) based on techniques used for *Achorripsis*
- System produces "score tables" that are transcribed into Western notation

JW= 1 A= 7.71 NA= 67 Q(I)=0.09/0.15/0.16/0.16/0.15/0.02/0.08/0.13/0.06/

N	TA	CLAS	INST	H	VIGL1	VIGL2	VIGL3	DUREE	DYNAM
1	0.	8	10	33.0	0.	0.	0.	0.	22
2	0.07	6	41	25.9	0.	0.	0.	13.94	54
3	0.09	9	1	60.7	0.	0.	0.	3.98	15
4	0.14	3	4	20.6	0.	0.	0.	0.89	1
5	0.24	3	2	50.1	0.	0.	0.	1.20	36
6	0.28	7	28	48.7	0.	0.	0.	0.	54
7	0.33	7	25	47.2	0.	0.	0.	0.	9
8	0.40	8	40	33.0	0.	0.	0.	0.	11
9	0.54	5	34	26.4	-8.0	-10.0	-6.0	4.72	53
10	0.68	8	38	24.1	0.	0.	0.	0.	54
11	0.72	2	5	42.0	0.	0.	0.	1.39	22
12	0.83	4	3	43.4	0.	0.	0.	1.60	15
13	0.85	2	4	58.3	0.	0.	0.	1.59	56
14	0.98	4	3	34.0	0.	0.	0.	1.76	55
15	1.23	4	2	42.0	0.	0.	0.	0.74	44
16	1.26	2	6	43.4	0.	0.	0.	2.12	15
17	1.28	3	2	61.9	0.	0.	0.	1.70	2
18	1.30	2	3	55.7	0.	0.	0.	0.29	38
19	1.34	2	3	58.1	0.	0.	0.	2.97	13
20	1.35	8	5	64.4	0.	0.	0.	0.	31
21	1.37	3	2	41.4	0.	0.	0.	0.	22
22	1.52	4	2	49.8	0.	0.	0.	0.02	53
23	1.59	5	32	46.7	-13.0	14.0	-11.0	4.10	25
24	1.63	7	28	44.7	0.	0.	0.	0.	7
25	1.68	6	38	41.5	0.	0.	0.	13.68	41
26	1.73	4	4	40.9	0.	0.	0.	0.66	13
27	1.73	2	6	18.4	0.	0.	0.	0.64	32
28	1.83	8	33	28.6	0.	0.	0.	0.	46
29	1.86	3	1	61.1	0.	0.	0.	0.79	14
30	1.95	2	3	40.1	0.	0.	0.	2.34	48
31	2.07	3	2	41.2	0.	0.	0.	1.21	9
32	2.19	1	4	0.	0.	0.	0.	8.63	56
33	2.33	5	16	47.8	-36.0	-24.0	-31.0	4.20	19
34	2.56	9	1	63.9	0.	0.	0.	1.84	54
35	2.61	5	22	67.6	-37.0	-50.0	31.0	12.97	41
36	2.67	8	46	23.4	0.	0.	0.	0.	33
37	2.75	4	1	67.9	0.	0.	0.	1.52	51
38	2.78	9	2	70.3	0.	0.	0.	6.06	6
39	2.92	4	4	25.1	0.	0.	0.	0.48	52
40	2.93	4	2	73.1	0.	0.	0.	1.02	25
41	2.98	7	42	25.9	0.	0.	0.	0.	43
42	3.08	4	2	54.7	0.	0.	0.	0.95	38
43	3.15	5	45	24.3	32.0	-20.0	26.0	5.78	60
44	3.17	5	43	38.4	21.0	-20.0	17.0	9.33	33
45	3.22	4	2	67.2	0.	0.	0.	0.34	60
46	3.22	8	41.	33.6	0.	0.	0.	0.	5
47	3.25	7	2	59.9	0.	0.	0.	0.	43
48	3.34	9	1	57.0	0.	0.	0.	2.50	47
49	3.52	1	7	0.	0.	0.	0.	17.06	10
50	3.67	8	13	54.3	0.	0.	0.	0.	41

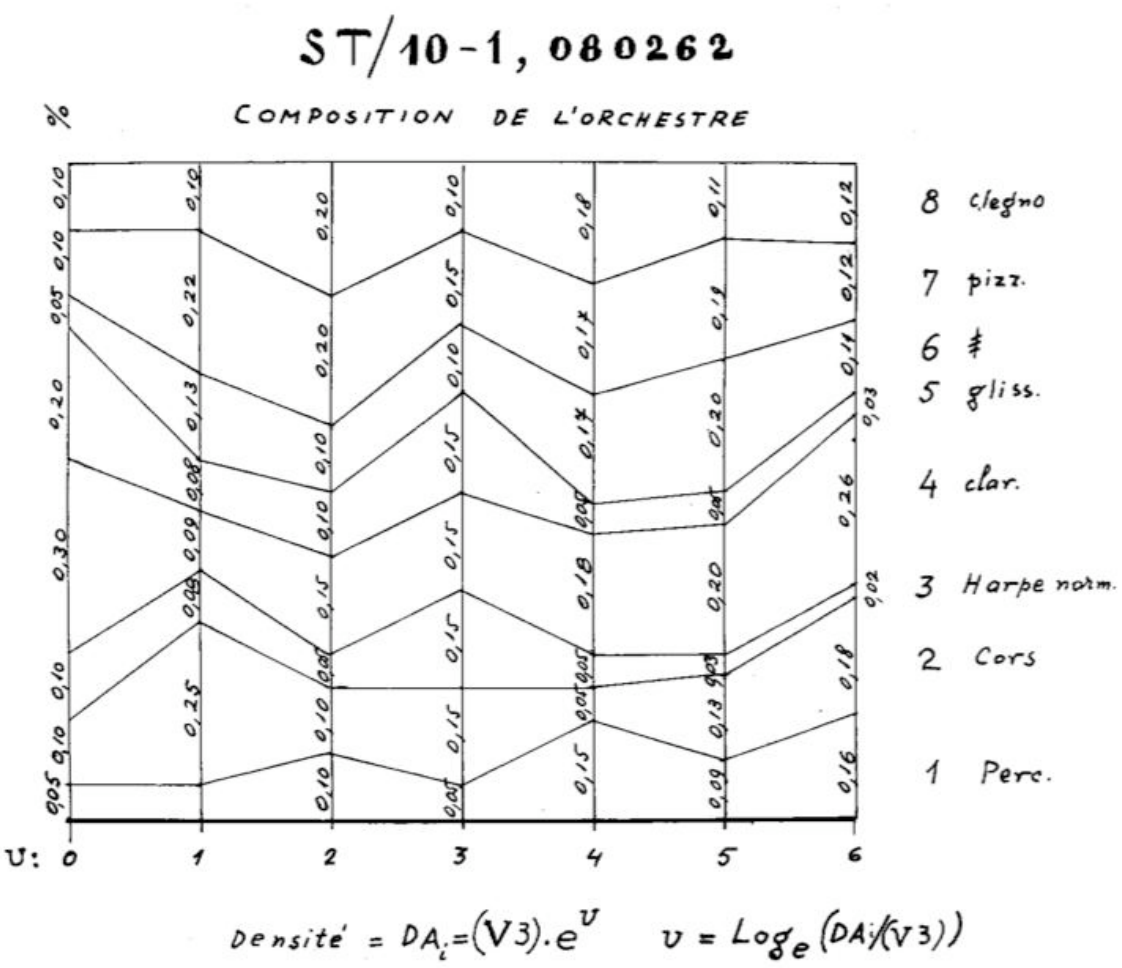
- 1962: *ST/10-1, 080262* (1956-1962) was premiered at IBM France
- Numerous related ST compositions were created
- 1965: Complete program, in Fortran, published and distributed (Xenakis 1965)

12.9. The Stochastic Music Program and Density

- Employed density as a compositional parameter at many levels
- Method

    1. Duration of each movement is determined
    2. The mean density of notes during a movement is calculated (in events per unit of time)

3. Percentage of events given to each timbre class is determined



Source: Xenakis (1971). © Scott Foresman/Pearson. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/fairuse.

4. For each event, the starting time point within the movement is calculated

5. From previously selected timbre classes, an instrument is chosen

6. A random chromatic pitch is chosen (as a shift of the instrument's previous note)

7. The duration of the note is determined based on an instrument-specific mean

8. The events dynamic contour is selected form a list of 44 options

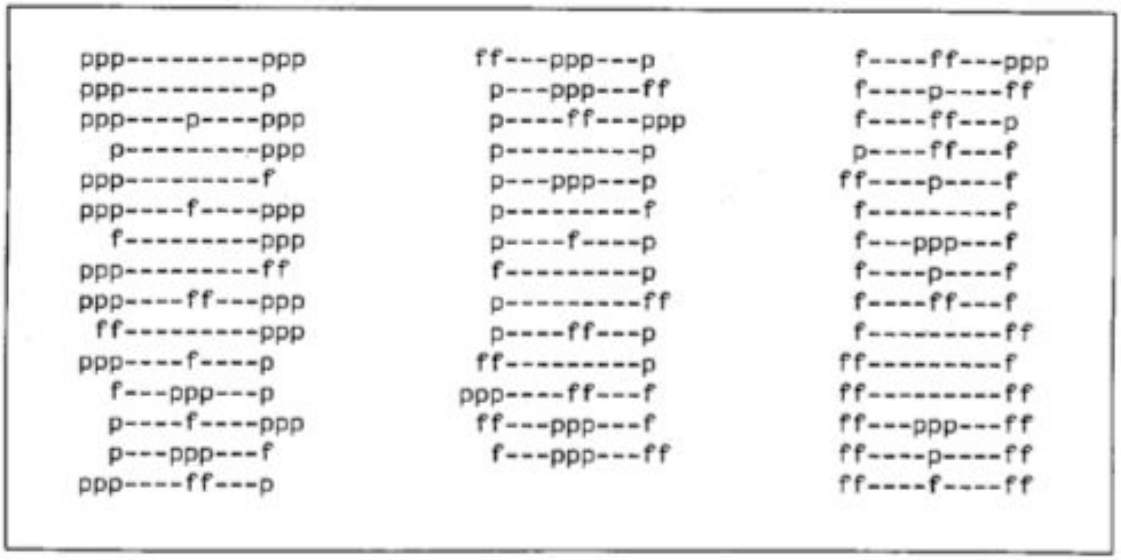


Fig. 56 Table of the 44 dynamic forms: a linear combination of 4 mean dynamic values, *ppp, p, f, ff*.

12.10. Listening: Xenakis

- Xenakis, ST-10, 1962
- Xenakis, Atrées, 1960

- Xenakis, ST-48, 1967
- Xenakis, ST-4

12.11. Reading: Xenaxis, Free Stochastic Music

- Xenakis, I. 1971. "Free stochastic Music." In *Cybernetics, art and ideas*. J. Reichardt, ed. Greenwich: New York Graphic Society. 124-142.
- Numerous publications include related/identical material
    - Xenakis, I. 1965. "Free Stochastic Music from the Computer. Programme of Stochastic music in Fortran." *Gravesaner Blätter* 26.
    - Xenakis, I. 1992. *Formalized Music: Thought and Mathematics in Music*. Indiana: Indiana University Press.

- How does Xenakis describe the public reaction to the use of computers in music?
- Xenakis describes mental mechanisms: are these just rules or mathematics?
- Xenakis imagines two new roles for contemporary composers: what are they?
- What are some of the advantages that Xenakis offers through the use of electronic brains?

12.12. Composing with Densities using TM TimeFill and a Noise Instrument

- TM LineGroove produces non-overlapping, linear events
- TM TimeFill will fill a time region with events, where position within the time span is determined by a ParameterObject
- Total number of events is determined by a ParameterObject
- Look at TM TimeSegment for a way to divide a texture into segments, each with independent fill densities
- Command sequence:
  - emo cn
  - tmo tf
  - tin a 80
  - tie t 0,30
  - *total event count is defined as static texture parameter, not a ParameterObject*

    tie s3 600
  - *start position within texture normalized within unit interval*

    tie d0 rb,.3,.3,0,1
  - *durations are independent of start time*

    tie r cs,(mv,a{.01}b{1.5}c{3}:{a=20|b=1|c=1})
  - *must reduce amplitudes*

    tie a ru,.5,.9
  - eln; elr; elh

12.13. Composing with Densities using TM TimeFill and a Single Sample

- Total number of events is determined by the combination of two ParameterObjects with IterateCross
- Command sequence:
  - emo cn
  - tmo tf
  - tin a 32
  - *set a file path to an audio file*

    tie x6 cf,/Volumes/xdisc/_sync/_x/src/martingale/martingale/audio/27980-high-slow.aif
  - *start position within audio file in seconds*

    tie x5 ru,0,1
  - *vary a low pass filter start and end frequencies*

    tie x2 mv,a{200}b{1000}c{10000}:{a=6|b=2|c=1}

    tie x3 mv,a{200}b{1000}c{10000}:{a=6|b=2|c=1}
  - *total event count is defined as static texture parameter, not a ParameterObject*

    tie s3 500
  - *start position within texture normalized within unit interval*

    tie d0 ic,(rg,.2,.1,0,1),(rg,.7,.1,0,1),(bg,rc,(0,1))
  - *durations are independent of start time*

    tie r cs,(whps,e,(bg,rp,(5,10,15)),0,.010,.100)
  - *must reduce amplitudes*

    tie a ru,.1,.3
  - eln; elr; elh

Chapter 13. Meeting 13, Approaches: Non-Standard Synthesis

13.1. Announcements

- Musical Design Report 3 due 6 April
- Start thinking about sonic system projects

13.2. The Xenakis Sieve

- A system (notation) for generating complex periodic integer sequences
- Described by Xenakis in at least six articles between 1965 and 1990
- Xenakis demonstrated application to pitch scales and rhythms, and suggested application to many other parameters
- "the basic problem for the originator of computer music is how to distribute points on a line" (Xenakis 1996, p. 150)
- "the image of a line with points on it, which is close to the musician and to the tradition of music, is very useful" (Xenakis 1996, p. 147)

13.3. The Xenakis Sieve: Basic Components

- Residual Classes: integer sequences based on a modulus (period) and a shift
  - Residual class 2@0: {..., 0, 2, 4, 6, 8, 10, 12, ...}
  - Residual class 2@1: {..., 1, 3, 5, 7, 9, 11, 13, ...}
  - Residual class 3@0: {..., 0, 3, 6, 9, 12, 15, ...}
- Sieves combine residual classes with logical operators
  - Sieve 3@0 | 4@0 : {..., 0, 3, 4, 6, 8, 9, 12, ...}
  - Sieve 3@0 & 4@0 : {..., 0, 12, 24, ...}
  - Sieve {-3@2&4} | {-3@1&4@1} | {3@2&4@2} | {-3@0&4@3}: {..., 0, 2, 4, 5, 7, 9, 11, 12, ...}
- Notation
  - Notations used by Xenakis:

$$\left(\overline{3}_{n+2} \cap 4_n\right) \cup \left(\overline{3}_{n+1} \cap 4_{n+1}\right) \cup \left(3_{n+2} \cap 4_{n+2}\right) \cup \left(\overline{3}_n \cap 4_{n+3}\right)$$

$$\{(3,2) \cap (4,7) \cap (6,11) \cap (8,7)\} \cup \{(6,9) \cap (15,18)\} \cup \{(15,5) \cap (8,6) \cap (4,2)\} \cup \{(6,9) \cap (15,19)\}$$

$$[(3,2) * (4,7)] + [(6,9) * (15,18)]$$

- A new notation (Ariza 2005c)

  Modulus number "at" shift value: 3@5

  Logical operators and (&), or (|), and not (-)

  Nested groups with braces: {-3@2&4} | {-3@1&4@1} | {3@2&4@2} | {-3@0&4@3}

13.4. An Object Oriented Implementation of the Sieve in Python

- sieve.py: a modular, object oriented sieve implementation in Python (Ariza 2005c)
- A low level, portable interface
-

```
>>> from athenaCL.libATH import sieve, pitchTools
>>> a = sieve.Sieve('{-3@2&4}|{-3@1&4@1}|{3@2&4@2}|{-3@0&4@3}')
>>> print a
{-3@2&4@0}|{-3@1&4@1}|{3@2&4@2}|{-3@0&4@3}
>>> a.period()
12
>>> a(0, range(0,13)) # one octave segment as pitch class
[0, 2, 4, 5, 7, 9, 11, 12]
>>> a.compress()
>>> print a
6@5|12@0|12@2|12@4|12@7|12@9
>>> a.expand()
>>> print a
{-3@2&4@0}|{-3@1&4@1}|{3@2&4@2}|{-3@0&4@3}
```

```
>>> a(0, range(0,12), 'wid')
[2, 2, 1, 2, 2, 2]
>>> a(0, range(0,12), 'bin')
[1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1]
>>> a(0, range(0,12), 'unit')
[0.0, 0.18181818181818182, 0.36363636363636365, 0.45454545454545453,
0.63636363636363635, 0.81818181818181823, 1.0]

>>> [pitchTools.psToNoteName(x) for x in a(0, range(49))]
['C4', 'D4', 'E4', 'F4', 'G4', 'A4', 'B4', 'C5', 'D5', 'E5', 'F5', 'G5', 'A5', 'B5',
'C6', 'D6', 'E6', 'F6', 'G6', 'A6', 'B6', 'C7', 'D7', 'E7', 'F7', 'G7', 'A7', 'B7',
'C8']
```

- sieve.py: SievePitch objects specialized for pitch space usage

```
>>> from athenaCL.libATH import sieve
>>> a = sieve.SievePitch('6@5|12@0|12@2|12@4|12@7|12@9,c2,c4')
>>> a()
[-24, -22, -20, -19, -17, -15, -13, -12, -10, -8, -7, -5, -3, -1, 0]
>>> [x + 60 for x in a()]
[36, 38, 40, 41, 43, 45, 47, 48, 50, 52, 53, 55, 57, 59, 60]
```

- athenaObj.py: can create an athenaCL Interpreter object to automate athenaCL commands

```
>>> from athenaCL.libATH import athenaObj
>>> ath = athenaObj.Interpreter()
>>> ath.cmd('tmo da')
>>> ath.cmd('pin a 5@3|7@2,c3,c8 4@2|6@3,c2,c4')
>>> ath.cmd('pih')
```

13.5. The Sieve in athenaCL: Interactive Command Line

- Using the interactive command-line, pitch sieves can be created, viewed, and deployed
- Comma-separated arguments for complete specification: sieveString, lowerBoundaryPitch, upperBoundaryPitch, originPitch, unitSpacing
- Example:

```
PIn a 5@3|7@2,c2,c4,c2,1
```

- Multiple sieve-based multisets can be defined
- Example:

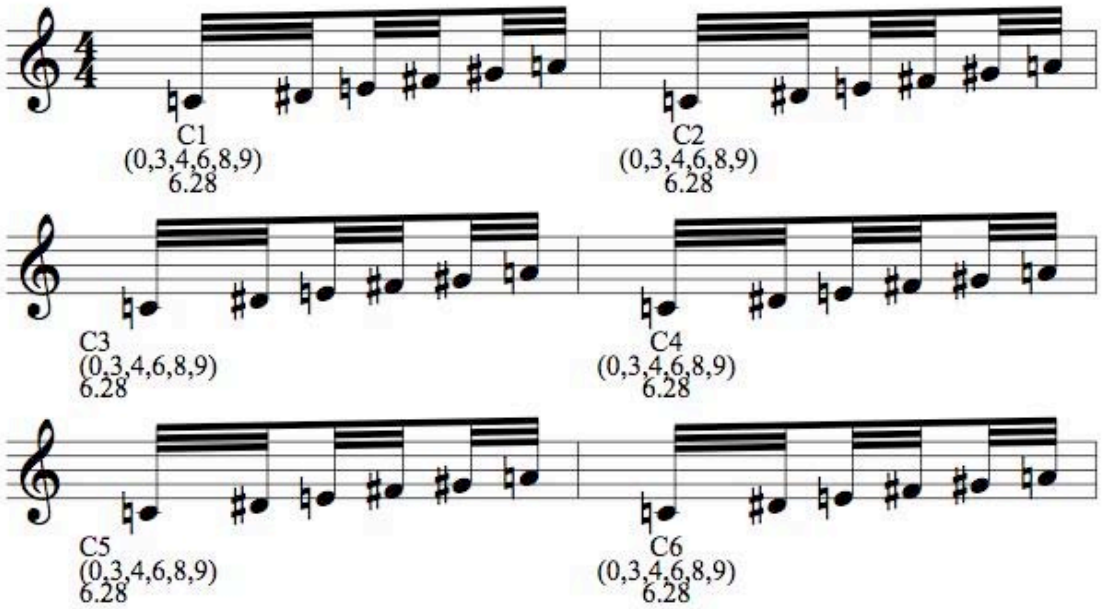
```
PIn b 5@3|7@2,c2,c4,c2,.5  5@1|7@8,c3,c6,c2,.5
```

13.6. Avoiding Octave Redundancy

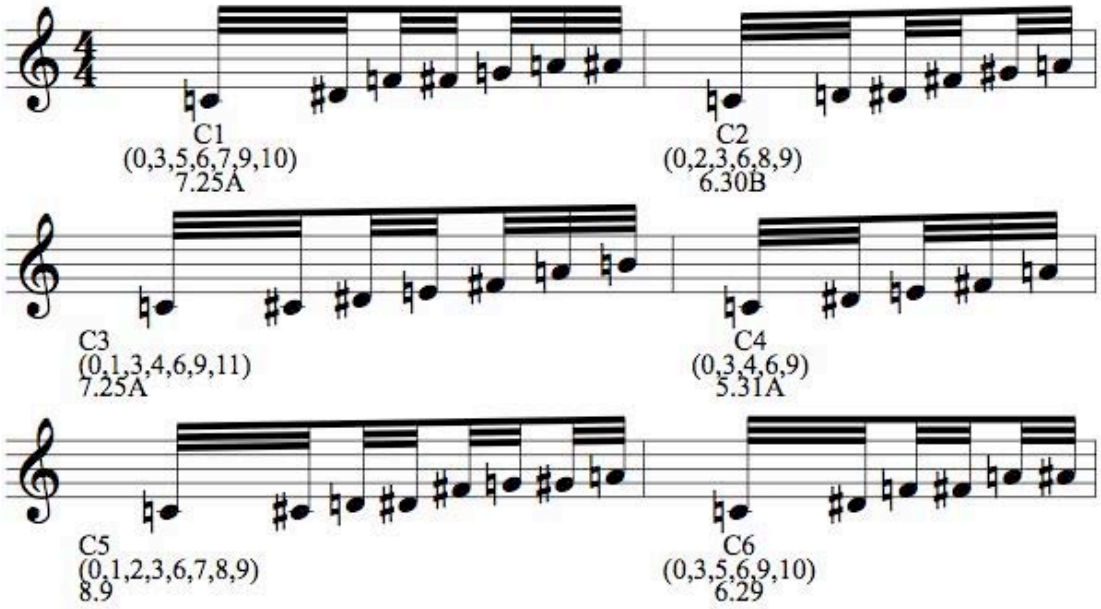
- Pitch sieves with large periods (or not a divisor or multiple of 12) are desirable
- Can be achieved simply through the union of two or more moduli with a high LCMs

```
>>> a = sieve.Sieve('3@0|4@0')Æ
```

```
>>> a.period()
12
```

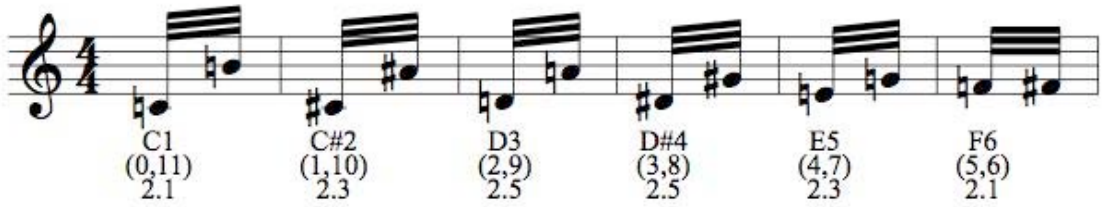



```
>>> a = sieve.Sieve('3@0|5@0|7@0')
>>> a.period()
105
```

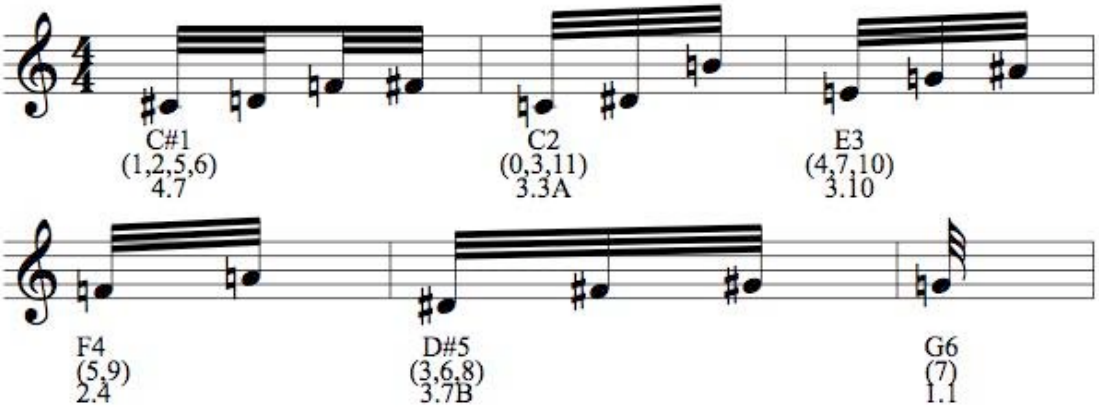


- Can be achieved through the use of moduli deviating from octave multiples (11, 13, 23, 25, 35, 37)

```
>>> a = sieve.Sieve('11@0|13@0')
>>> a.period()
143
```




```
>>> a = sieve.Sieve('11@1|13@2|23@5|25@6')
>>> a.period()
82225
```



13.7. Deploying Pitch Sieves with HarmonicAssembly

- Provide complete sieve over seven octaves
- TM HarmonicAssembly used to create chords
- Chord size randomly selected between 2 and 3
- Rhythms and rests created with zero-order Markov chains
- Command sequence:
  - emo m

- pin a 11@1|13@2|23@5|25@6,c1,c7
- tmo ha
- tin a 0
- tie t 0,30
- tie a rb,.2,.2,.6,1
- tie b c,120
- *zero-order Markov chains building pulse triples*

  tie r pt,(c,4),(mv,a{1}b{3}:{a=12|b=1}),(mv,a{1}b{0}:{a=9|b=1}),(c,.8)
- *index position of multiset: there is only one at zero*

  tie d0 c,0
- *selecting pitches from the multiset (indices 0-15) with a tendency mask*

  tie d1 ru,(bpl,t,l,[(0,0),(30,12)]),(bpl,t,l,[(0,3),(30,15)])
- *repetitions of each chord*

  tie d2 c,1
- *chord size*

  tie d3 bg,rc,(2,3)
- eln; elh

13.8. Reading: Berg. Composing Sound Structures with Rules

- Berg, P. 2009. "Composing Sound Structures with Rules." *Contemporary Music Review* 28(1): 75-87.
- How did the PDP-15 affect what techniques were explored at the Institute of Sonology
- Given the music, find the rules: how is this different than analytical approaches?
- What is non standard about non-standard synthesis?
- What is the relationship of Berg's ASP to PILE

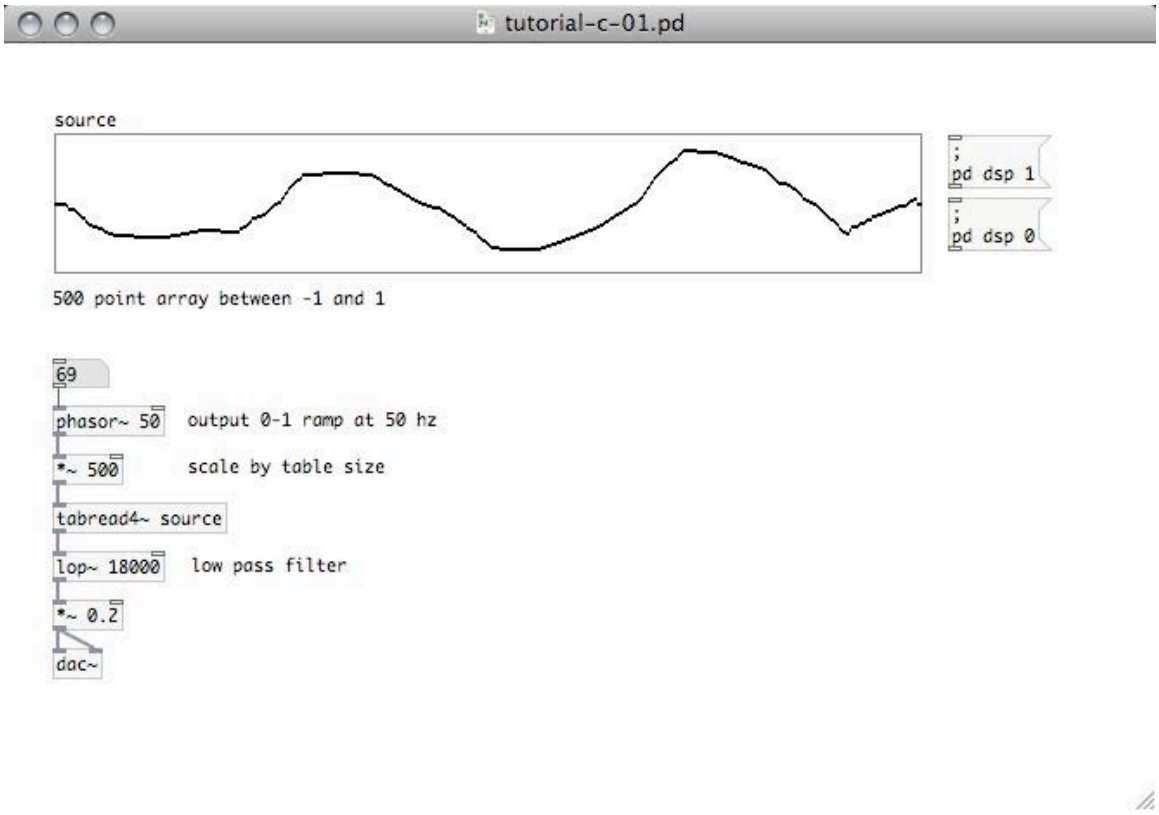
13.9. Non-Standard Synthesis: Xenakis and Koenig

- Both began with techniques for creating score tables
- Both explored apply this techniques to sound construction
- Both rejected acoustic models of sound creation
- Both employed techniques of dynamic, algorithmic waveform generation

13.10. Tutorial: a Dynamic Stochastic Wavetable

- Looping through an array at the audio rate creates a wave table

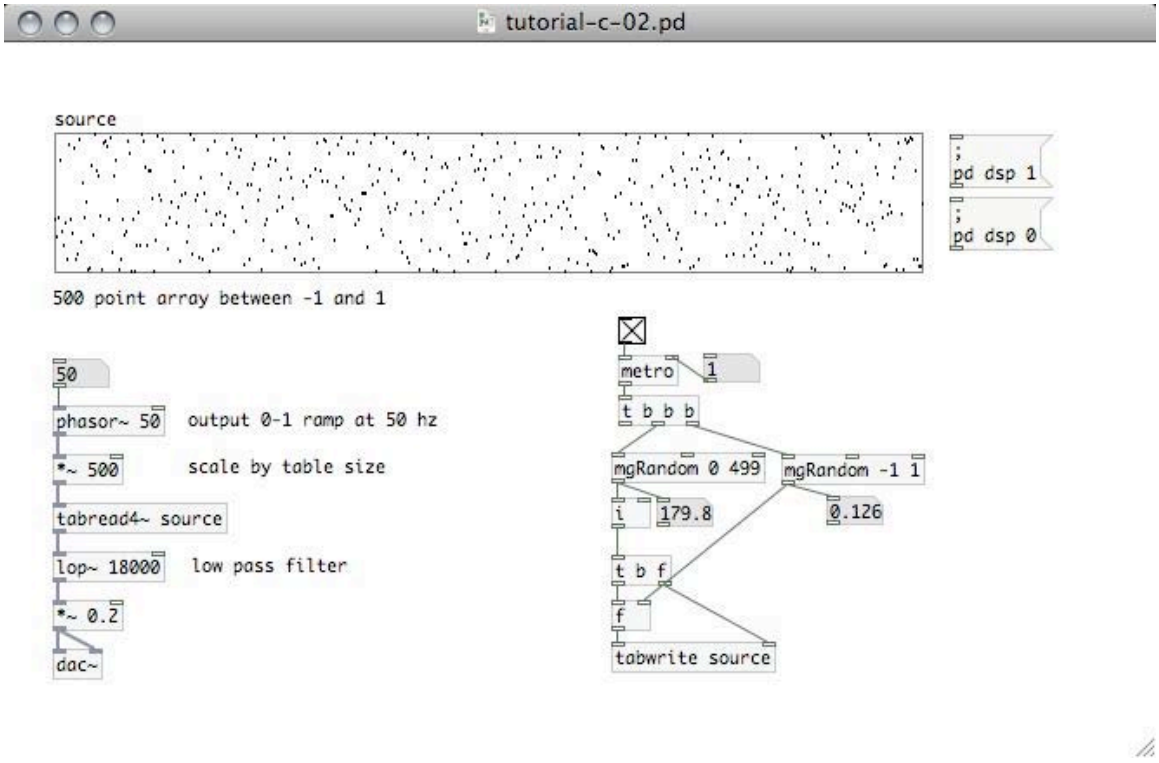
  [tabread4~] interpolates between points for any [phasor~] rate



- Randomly place points within the table at a variable rate controlled by a [metro]

  [tabwrite] lets us specify index position, value to write to

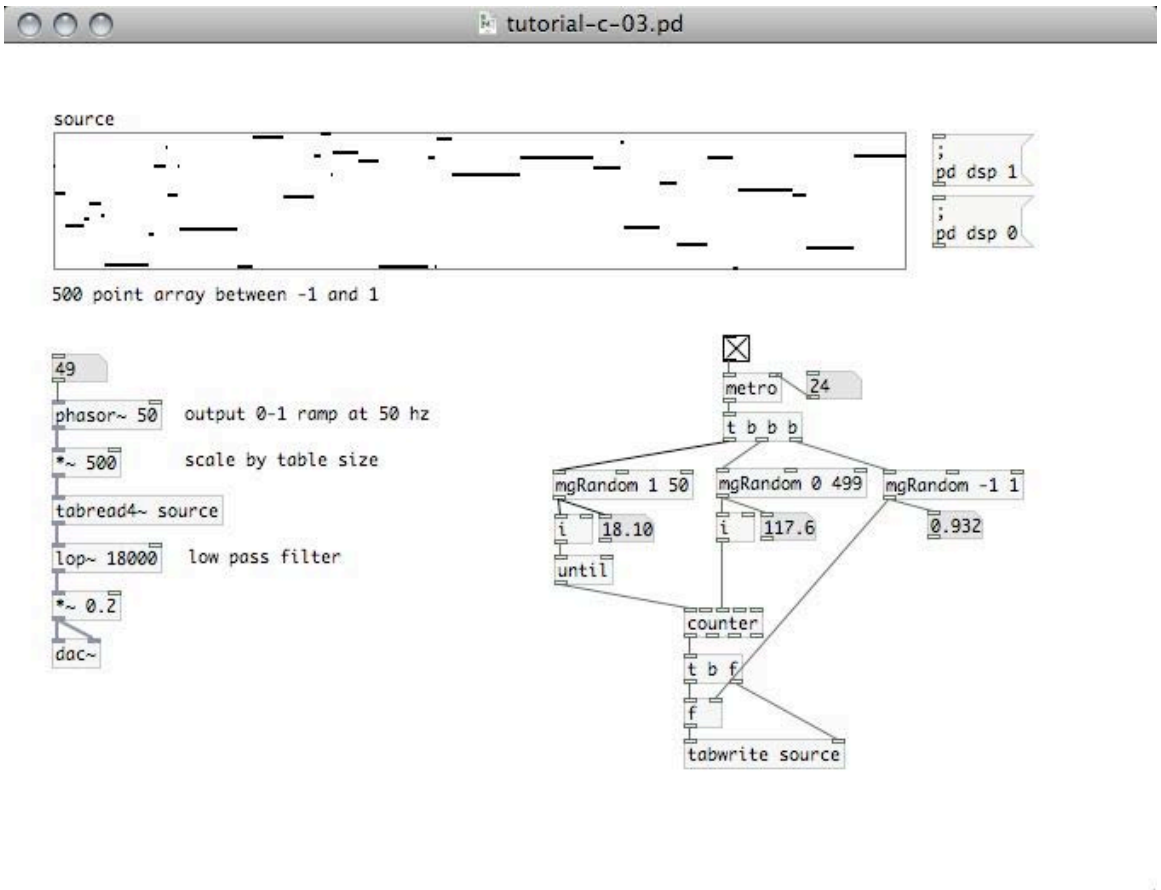
  Can only be done at the event rate (1 ms updates)



- Randomly draw line segments instead of points

  [until] will send out as many bangs as provided as an argument

  [counter] can receive a new minimum to designate start index for each segment generation

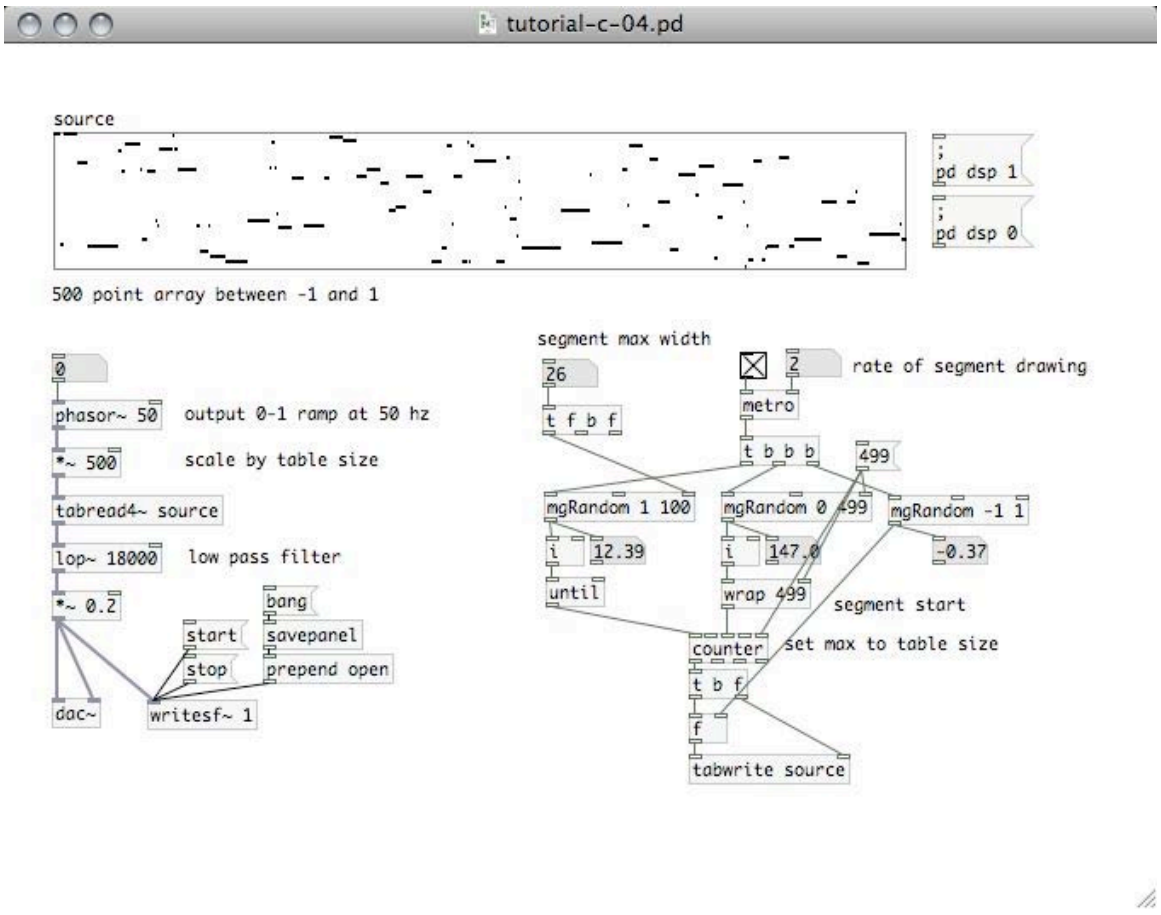


- Randomly draw line segments instead of points

  Use [wrap] to ensure points stay within table

  Set max of [counter] table

  Record output to a new file with [writesf~]



13.11. Non-Standard Synthesis: Xenakis and Koenig

- Both began with techniques for creating score tables
- Both explored apply this techniques to sound construction
- Both rejected acoustic models of sound creation
- Both employed techniques of dynamic, algorithmic waveform generation

13.12. Koenig: SSP

- Application of Koenig's selection principles to waveforms
- Proposed in 1972, implemented in 1977
- Given a collection of discrete time and amplitude values, select from these to create waveform break points
- Program was conversational, interactive

- Use of tendency masks to control segment generation produced directly audible results (Berg 2009, p. 84)

13.13. Xenakis: GENDYN

- Dynamic Stochastic Synthesis
- Explored by Xenakis over many years, starting in the 1970s
- Not based on natural or acoustical models of sound
- Algorithmically create waveforms by generating time and amplitude coordinates with second order random walks, then interpolating to create wave forms

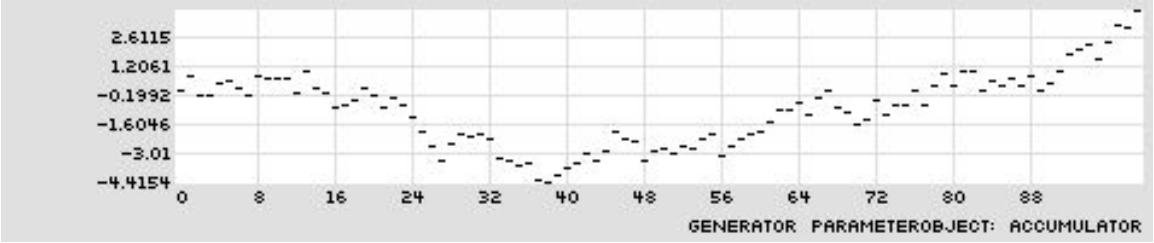
13.14. Reading: Hoffman. A New GENDYN Porgram

- Hoffman, P. 2000. "A New GENDYN Program." *Computer Music Journal* 24(2): 31-38.
- Hoffman describes GENDYN as a "rigrous algorithmic composition procedure"; what does he mean? Is he correct?
- How did Xenakis compose, at the largest scale, with GENDYN?
- What does Hoffman say about Xenakis's programming style?

13.15. Second-Order Random Walks as ParameterObjects

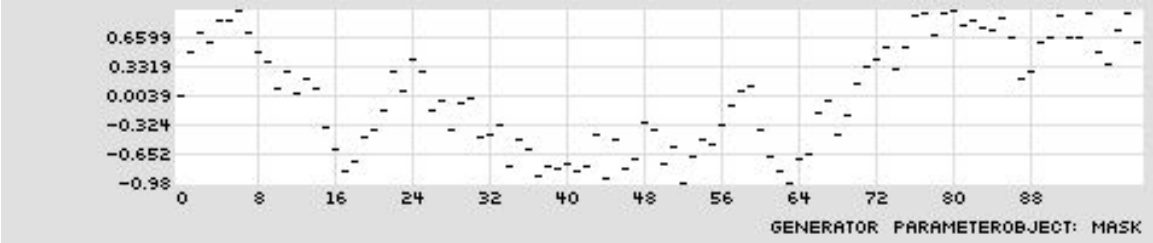
- Accumulator: permit consecutively summing values based on an initial value and the output of a ParameterObject

```
:: tpmap 100 a,0,(ru,-1,1)
accumulator, 0, (randomUniform, (constant, -1), (constant, 1))
TPmap display complete.
```



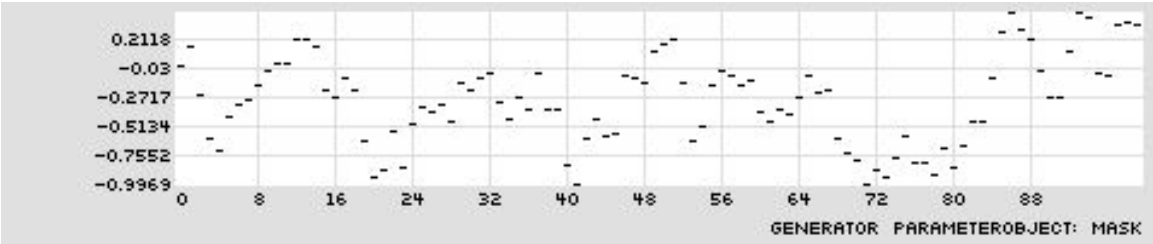
- Mask:Êconstrain the output of a ParameterObject within boundaries created by two ParameterObjects; boundaries can be limit, wrap, or reflect

```
:: tpmap 100 mask,reflect,(c,-1),(c,1),(a,0,(ru,-.5,.5))
mask, reflect, (constant, -1), (constant, 1), (accumulator, 0, (randomUniform,
(constant, -0.5), (constant, 0.5)))
TPmap display complete.
```



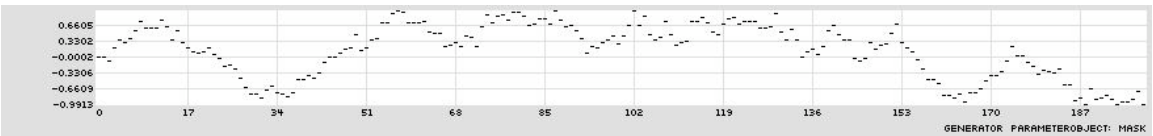
- Second order random walk: use (discrete) random walks to control the step size of another random walk

```
:: tpmap 100 m,r,(c,-1),(c,1),(a,0,(ru,(bg,rw,(-.1,-.2,-.3,-.4,-
.5)),(bg,rw,(.1,.2,.3,.4,.5))))
mask, reflect, (constant, -1), (constant, 1), (accumulator, 0, (randomUniform,
(basketGen, randomWalk, (-0.1,-0.2,-0.3,-0.4,-0.5)), (basketGen, randomWalk,
(0.1,0.2,0.3,0.4,0.5))))
TPmap display complete.
```



- Second order random walk: use (continuous) random walk to control the step size of another random walk

```
:: tpmap 200 m,r,(c,-1),(c,1),(a,0,(ru,(m,r,(c,-.5),(c,0),(a,0,(ru,-
.5,0))),(m,r,(c,0),(c,.5),(a,0,(ru,0,.5)))))
mask, reflect, (constant, -1), (constant, 1), (accumulator, 0, (randomUniform,
(mask, reflect, (constant, -0.5), (constant, 0), (accumulator, 0,
(randomUniform, (constant, -0.5), (constant, 0)))), (mask, reflect, (constant,
0), (constant, 0.5), (accumulator, 0, (randomUniform, (constant, 0), (constant,
0.5))))))
TPmap display complete.
```

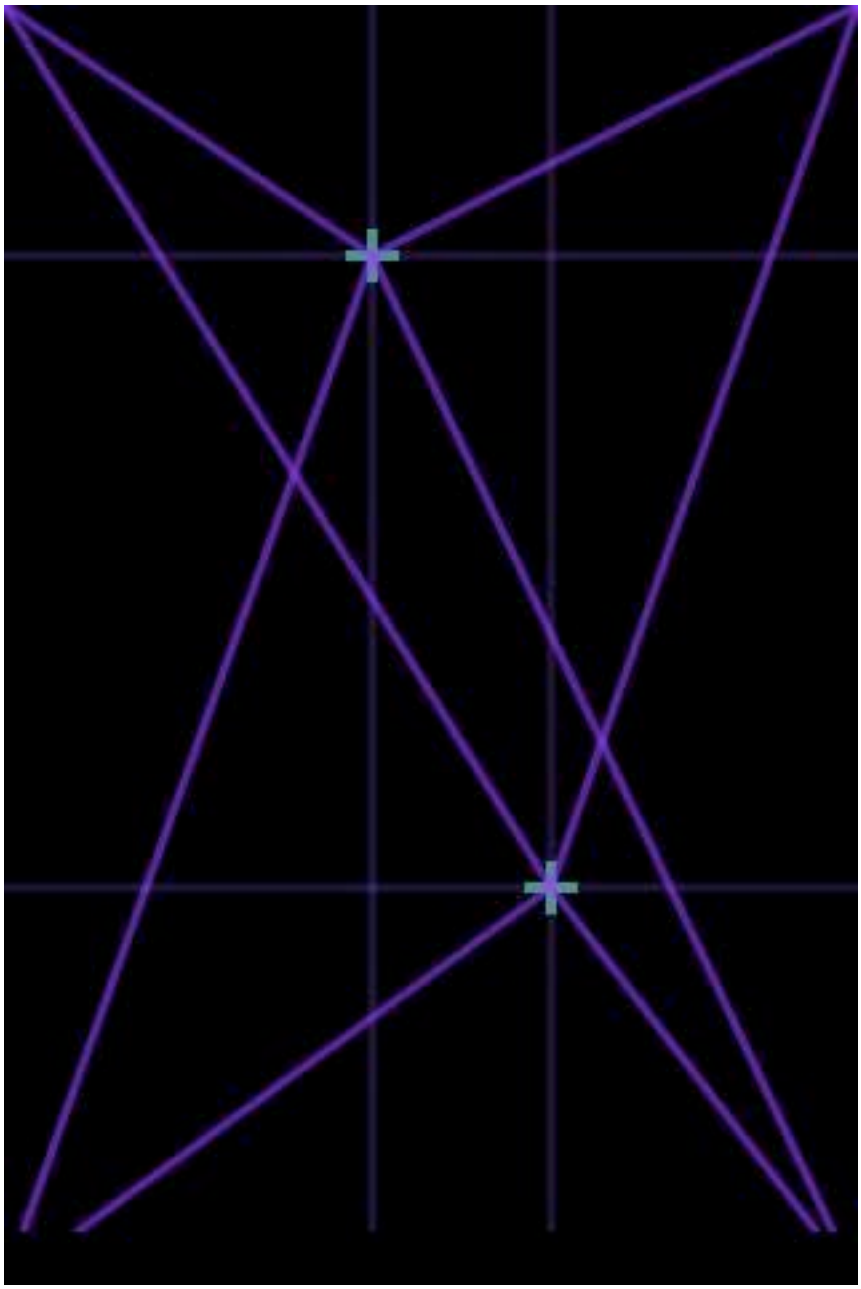


13.16. Listening: Xenakis

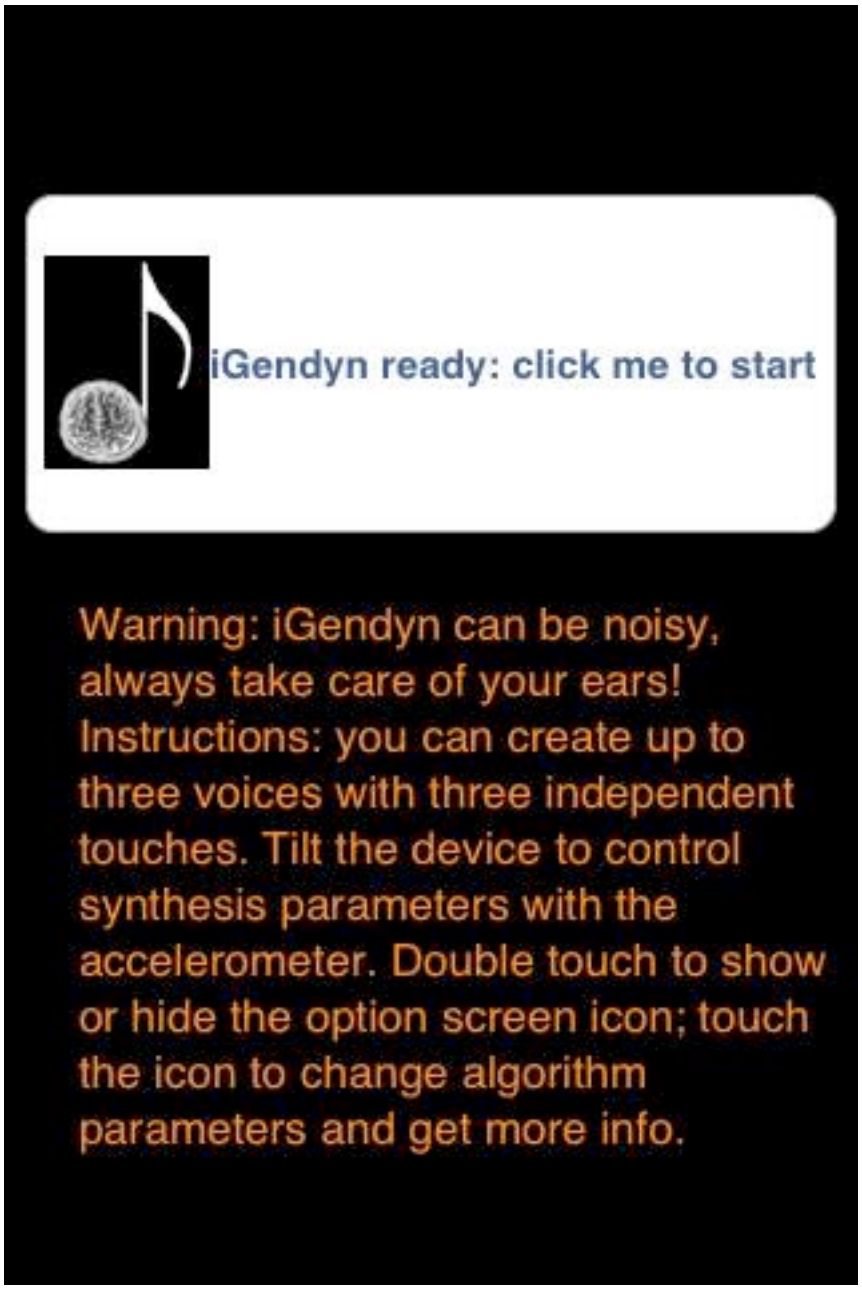
- Audio: S.709
- BBC interview with Xenakis on S.709
- "Music is not a language. Every musical piece is like a highly complex rock with ridges and designs engraved wtihin and without, that can be interpreted in a thousand ways without a single one being the best or the most true." (Xenakis 1987, p. 32)

13.17. iGendyn: Gendyn as iPhone / iPod touch App

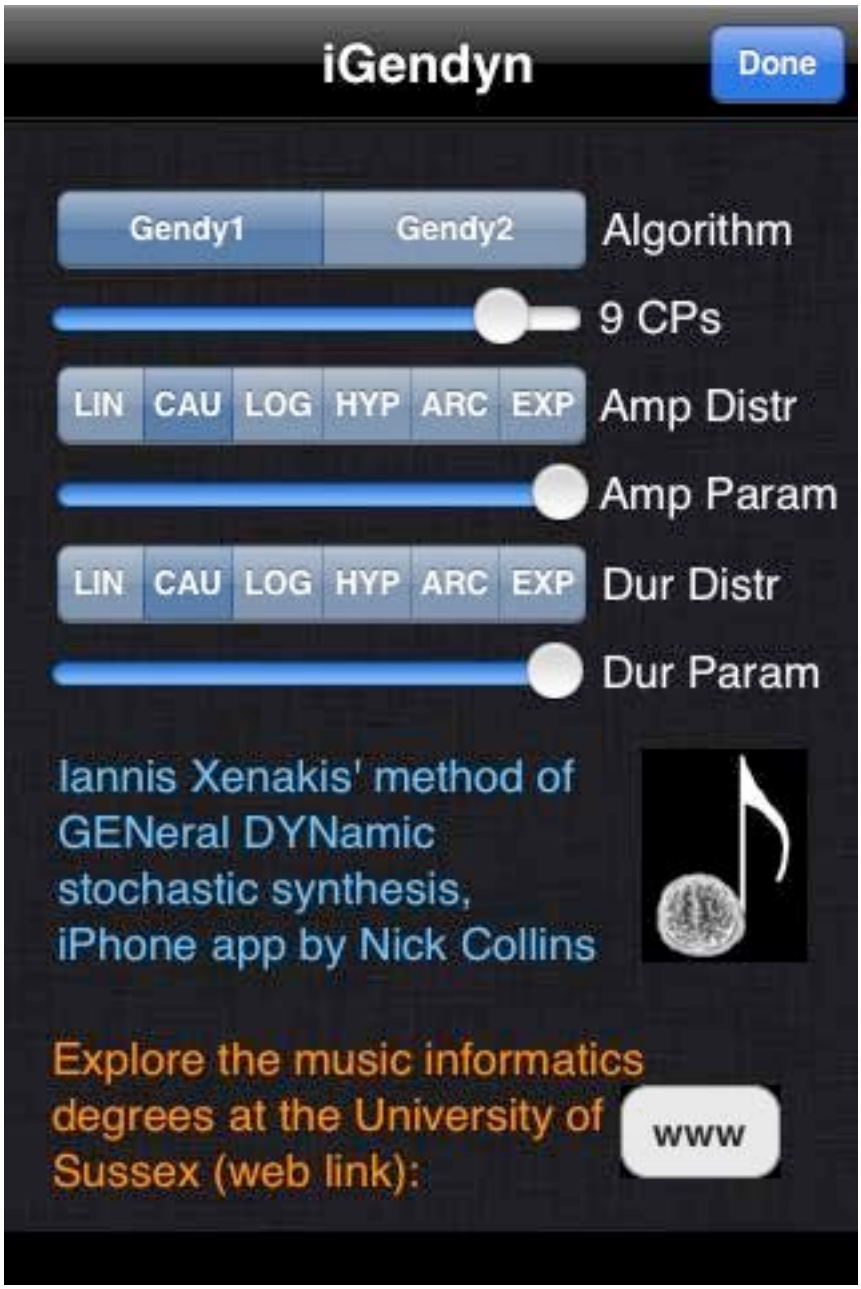
- Created by Nick Collins



Courtesy of Nick Collins. Used with permission.



Courtesy of Nick Collins. Used with permission.



Courtesy of Nick Collins. Used with permission.

Chapter 14. Meeting 14, Approaches: Granular and Concatenative Synthesis

14.1. Announcements

- Musical Design Report 3 due 6 April
- Sonic system draft due: 27 April
- Next couple of weeks: need to meet with me to talk about sonic system projects
- Quiz on Thursday

14.2. Musical Design Report 3

- Must be primarily built with rendered digital audio, such as output from Csound, PD, or related tools
- Density, and contrasts in density, must be a significant compositional parameter
- Must feature granular, concatenative, or sound montage synthesis techniques in some manner
- Should have at least one transition between disparate material that is a gradual morph, fade, or dove-tail
- Can be composed with athenaCL, athenaCL and other tools, or other tools alone
- Mixing audio obtained from PD and/or athenaCL/Csound in Audacity or a DAW is highly recommended.

14.3. Listening: Vaggione

- Listening: Horacio Vaggione, *24 Variations*, 2002

14.4. Reading: Roads, Introduction to Granular Synthesis

- Roads, C. 1988. "Introduction to Granular Synthesis." *Computer Music Journal* 12(2): 11-13.
- What are some common duration ranges and grains per second used in granular synthesis?
- Gabor's quanta

- Xenakis's screens and books of screens

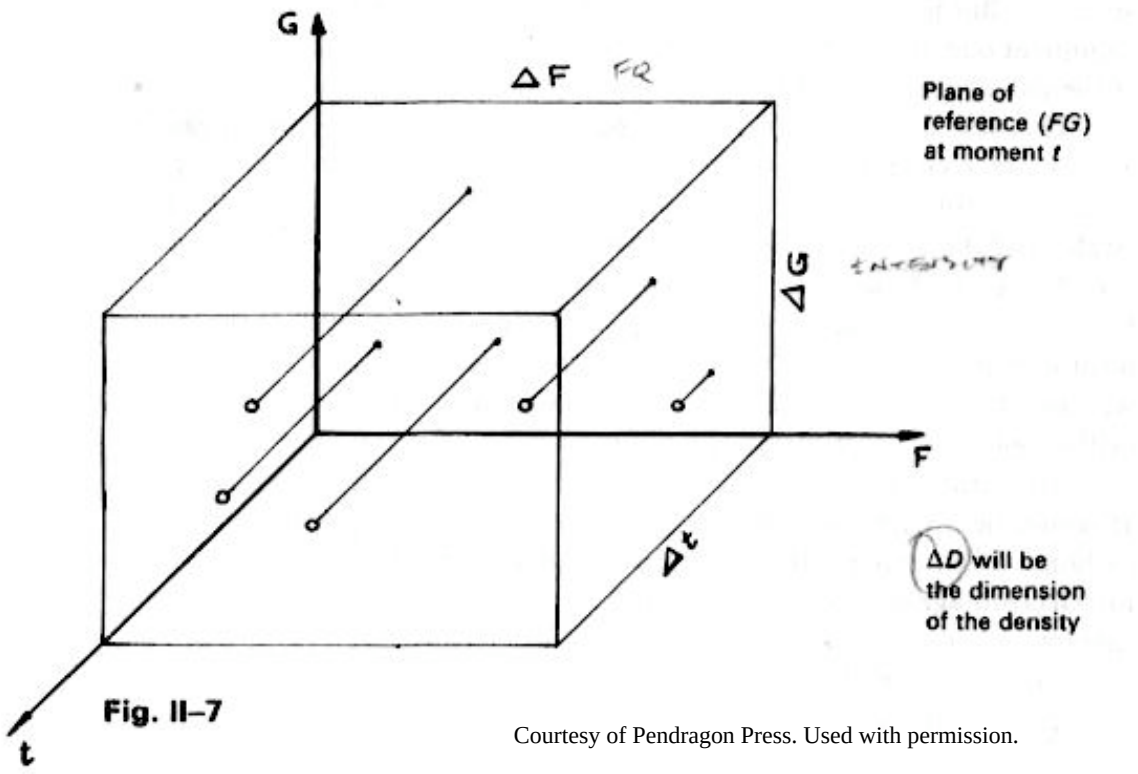


Fig. II–7

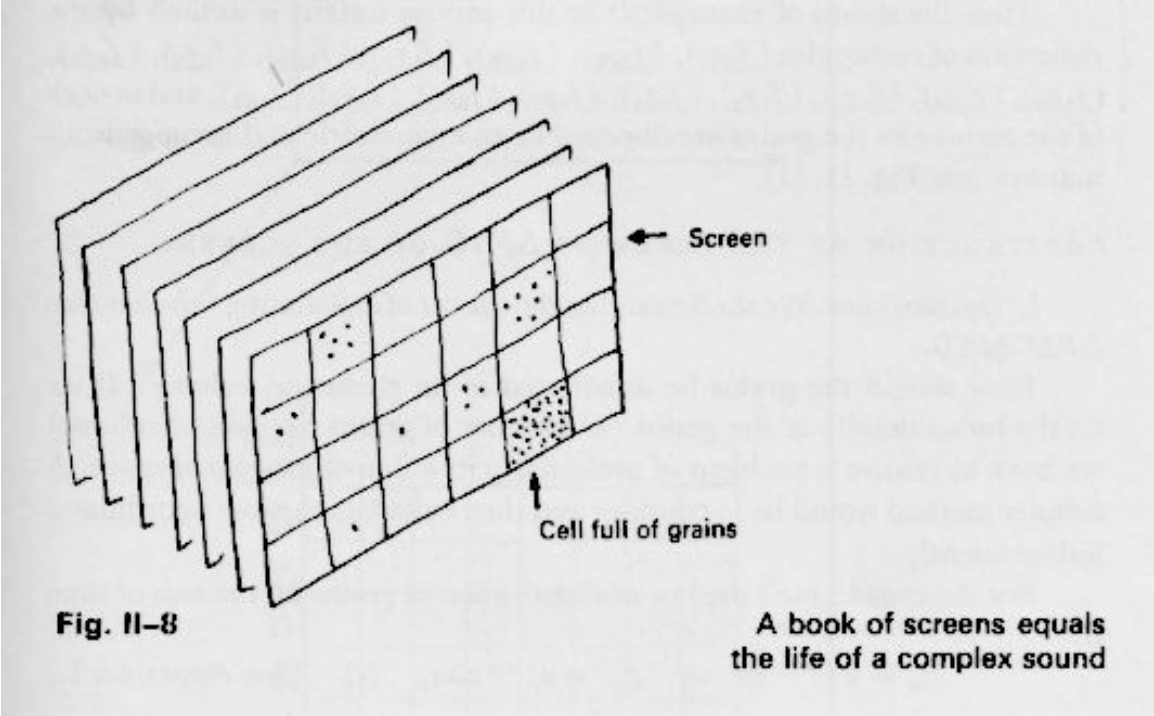


Fig. II–8

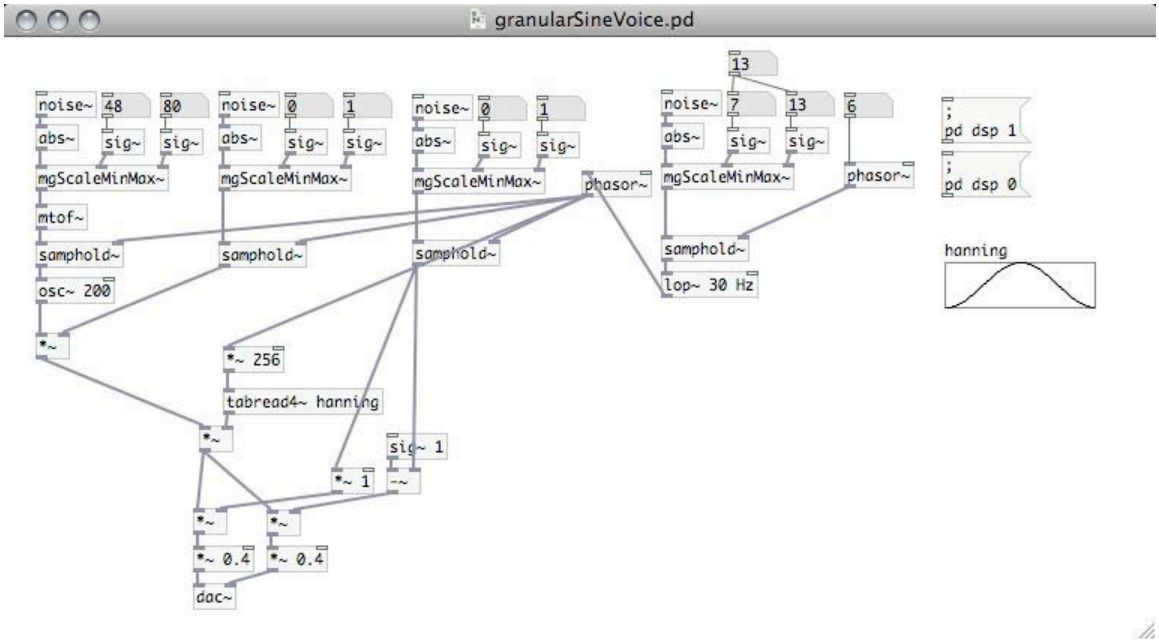
A book of screens equals the life of a complex sound

- What were some parameters that Roads employs in his implementation?

- What are some other applications of granular synthesis?
- What are the visual or animation analogues of granular synthesis?
- Is granular synthesis algorithmic composition?

14.5. Simple Sine Grains in PD

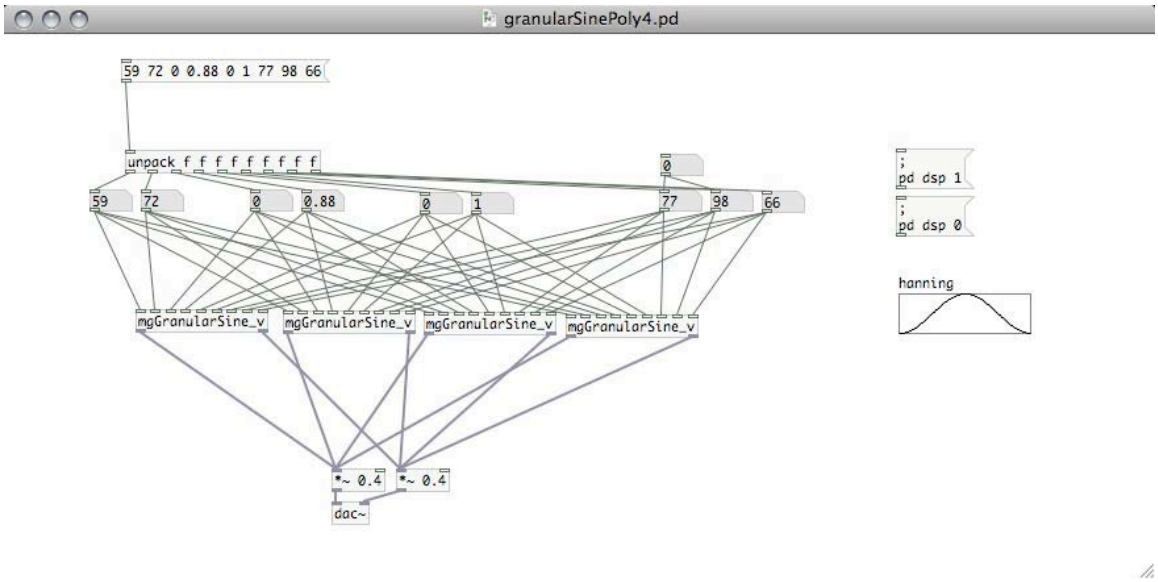
- Enveloped sine tines
- A [phasor~] is used to trigger multiple [samphold~] processes that grab parameter values once per event envelope
- Random parameter values are generated by [noise~] scaled between minimum and maximum values with [mgScaleMinMax~]
- Event envelopes are provided by the hanning array and read with [tabread4~]
- martingale/pd/demo/granularSineVoice.pd



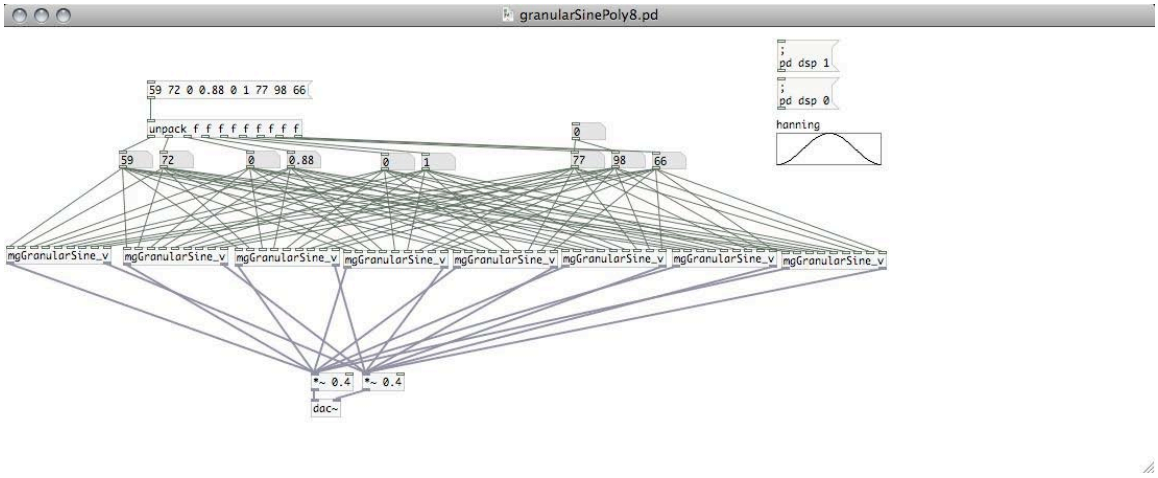
14.6. Polyphonic Sine Grains in PD

- Numerous instances of [mgGranularSine_v.pd] can be controlled together to produce multiple streams of grains
- Due to use of random parameter ranges, each voice will be independent

- martingale/pd/demo/granularSinePoly4.pd

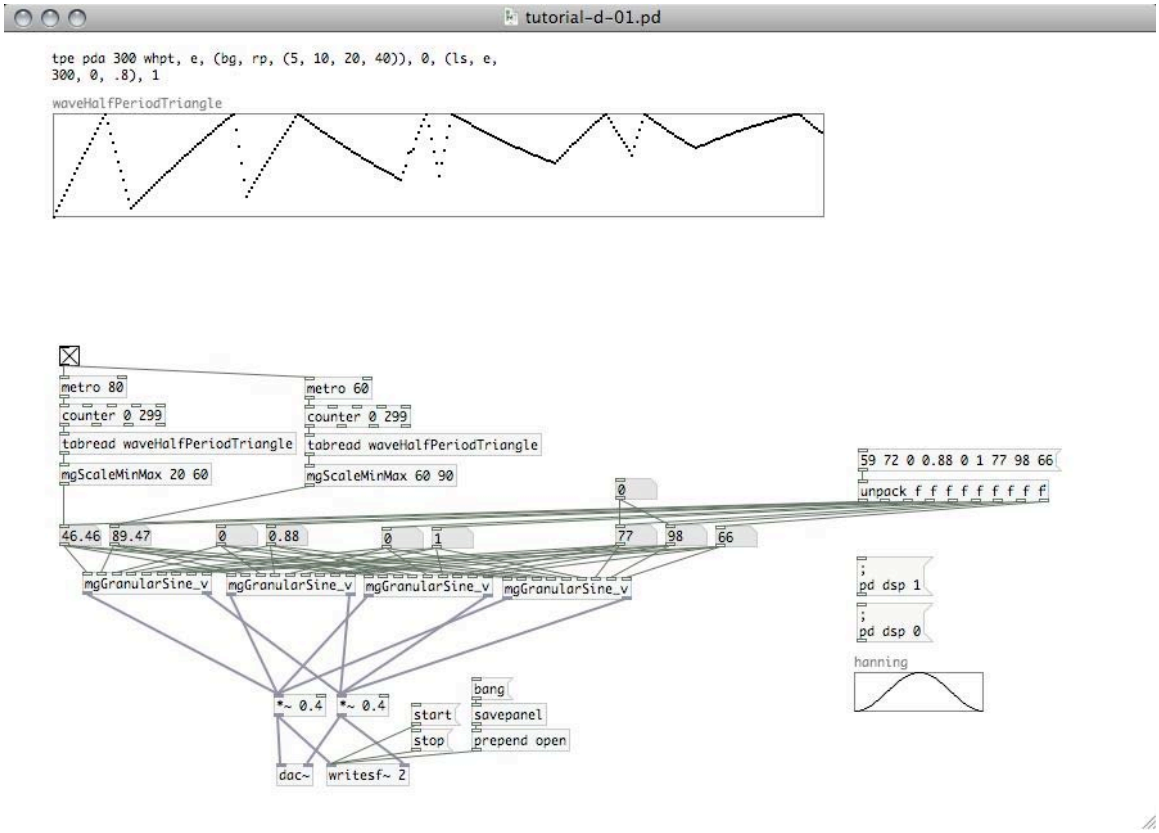


- martingale/pd/demo/granularSinePoly8.pd

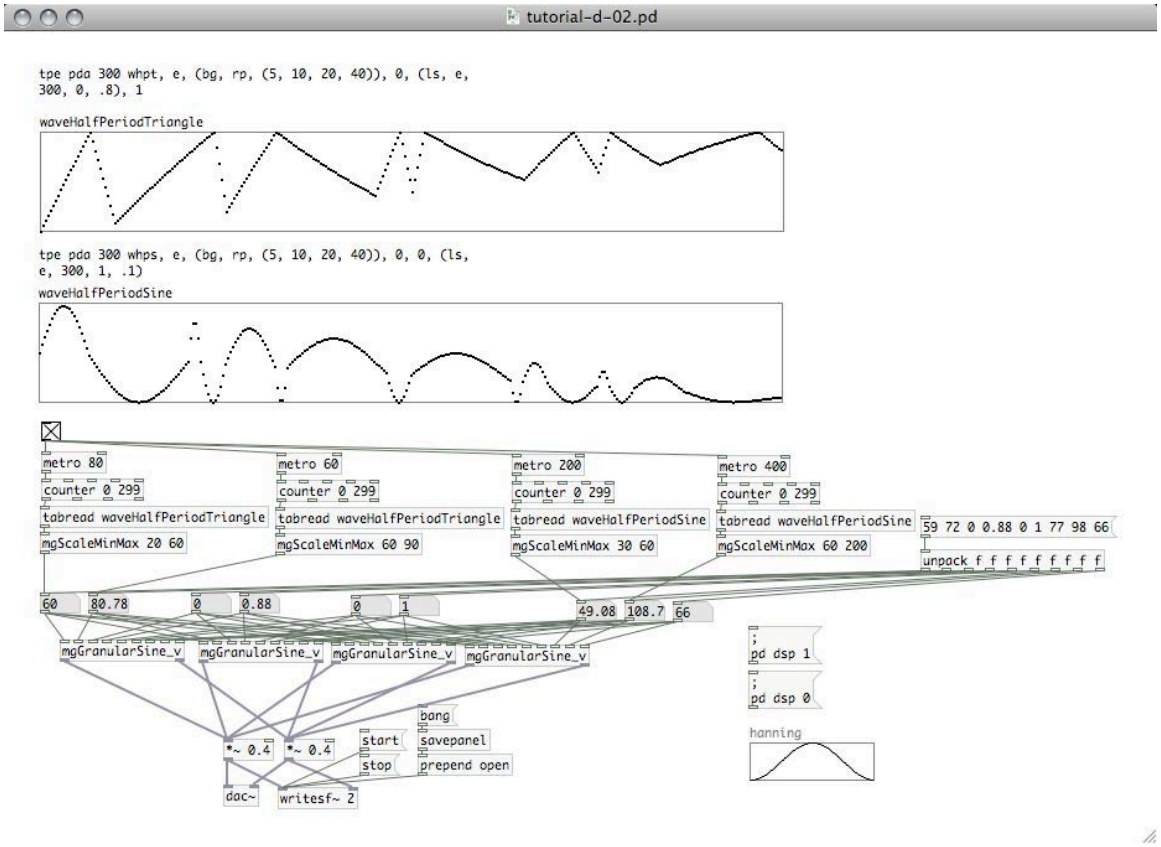


14.7. Large-Scale Parameter Behavior of Polyphonic Sine Grains in PD

- Use the TPe (TextureParameter Export) command with the PureDataArray format to create array structures
  - tpe pda 300 whpt,e,(bg,rp,(5,10,20,40)),0,(ls,e,300,0,.8),1
  - tpe pda 300 whps,e,(bg,rp,(5,10,20,40)),0,0,(ls,e,300,1,.1)
- Reading parameter values from multiple [tabread] at different rates



- Reading parameter values from multiple [tabread] and multiple tables at different rates



14.8. Polyphonic Sine Grains in athenaCL: LineGroove

- Can approach granular synthesis by using extremely small durations and/or fast tempi
- Command sequence using TM LineGroove:
  - emo cn
  - tmo LineGroove
  - tin a 4
  - *set a event time between 60 and 120 ms*

    tie r cs,(ru,.060,.120)
  - *smooth envelope shapes*

    tie x0 c,.1; tie x1 c,.5
  - *set field with a tendency mask converging on a single pitch after 15 seconds*

    tie f ru,(ls,t,15,-24,0),(ls,t,15,24,0)

- *set random panning*

  tie n ru,0,1

- *create a few more instances*

  ticp a b c d e f

- eln; elr; elh

14.9. Polyphonic Sine Grains in athenaCL: DroneArticulate

- TM DroneArticulate realizes each component of the path as a separate lines, writing an independent voice for each pitch one at a time for the entire duration
- Command sequence using TM DroneArticulate:
  - emo cn
  - tmo DroneArticulate
  - *a very large pitch collection made from a Xenakis sieve*

    pin a 5@2|7@6,c1,c9

  - tin a 4
  - *set a event time between 60 and 120 ms*

    tie r cs,(ru,.060,.120)

  - *smooth envelope shapes*

    tie x0 c,.1; tie x1 c,.5

  - *set random panning*

    tie n ru,0,1

  - *reduce amplitudes*

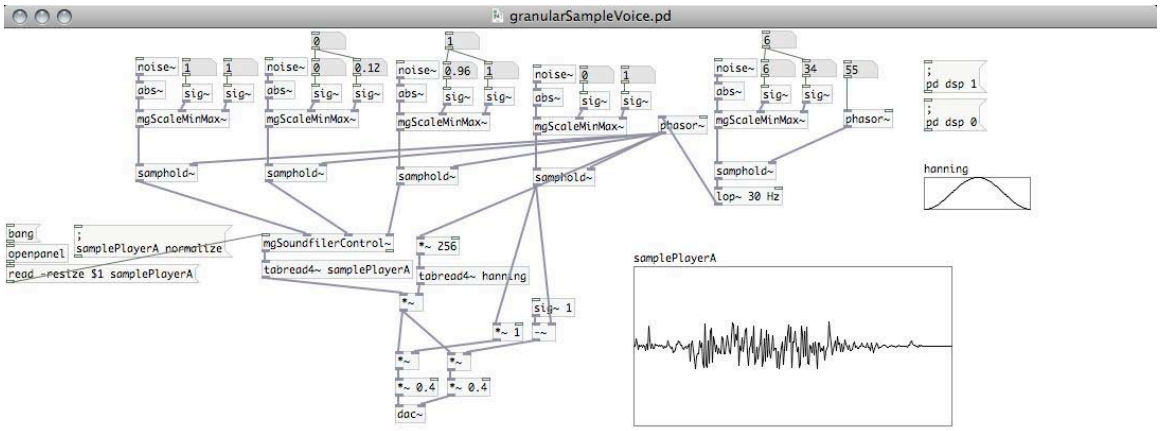
    tie a ru,.6,.8

  - eln; elr; elh

14.10. Simple Sample Grains in PD

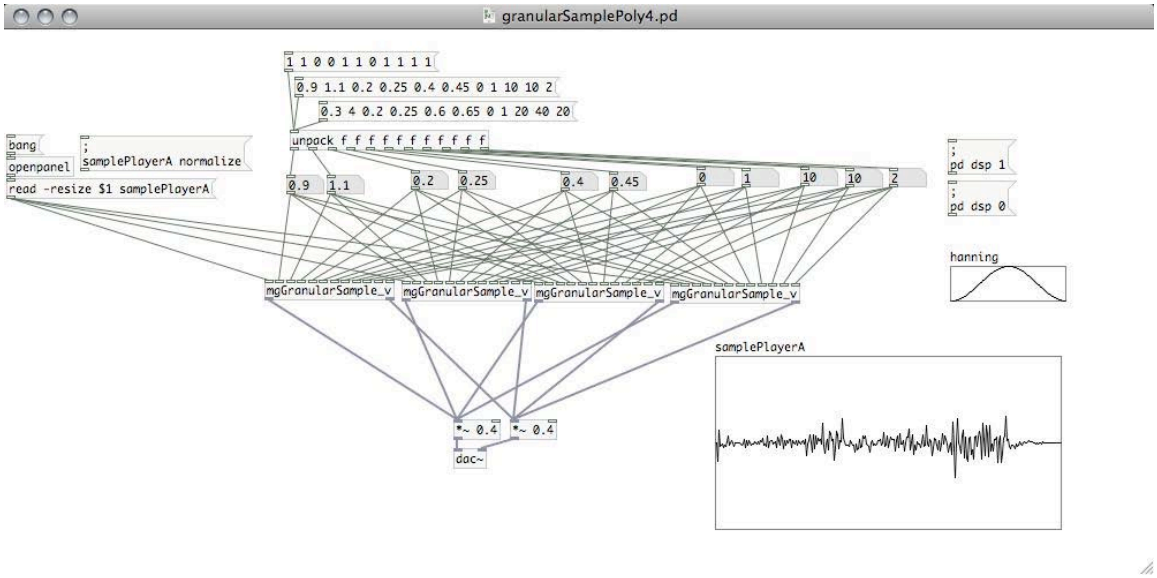
- Enveloped sampled audio files

- Press [bang] to trigger [openpanel] to select an audio file
- Parameters: playback speed min/max, start time min/max (within unit interval), end time min/max, pan min/max, phasor fq min/max, control phasor
- martingale/pd/demo/granularSampleVoice.pd



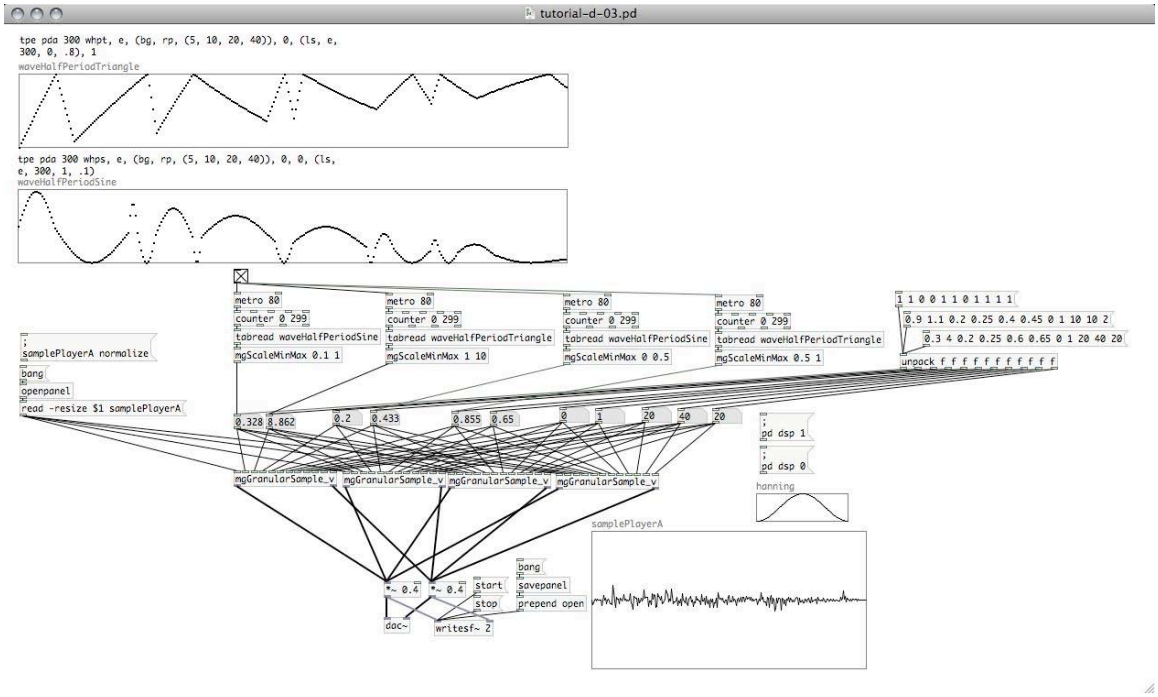
14.11. Polyphonic Sample Grains in PD

- Numerous instances of [mgGranularSample_v.pd] can be controlled together to produce multiple streams of grains
- martingale/pd/demo/granularSamplePoly4.pd



14.12. Large-Scale Parameter Behavior of Polyphonic Sample Grains in PD

- Use the TPe (TextureParameter Export) command with the PureDataArray format to create array structures
  - tpe pda 300 whpt,e,(bg,rp,(5,10,20,40)),0,(ls,e,300,0,.8),1
  - tpe pda 300 whps,e,(bg,rp,(5,10,20,40)),0,0,(ls,e,300,1,.1)
- Reading parameter values from multiple [tabread] and multiple tables at different rates



14.13. Polyphonic Sample Grains in athenaCL from a Single Audio File: LineGroove

- Read segments from an audio file by specifying the audio file (with the ConstantFile PO) and a start time
- Command sequence:
  - emo cn
  - tmo LineGroove
  - *instrument 32 is a fixed playback rate sample player*

tin a 32

- *set a file path to an audio file*

  tie x6 cf,/Volumes/xdisc/_sync/_x/src/martingale/martingale/audio/32673.aif

- *set a event time between 60 and 120 ms*

  tie r cs,(ru,.060,.120)

- *smooth envelope shapes*

  tie x0 c,.01; tie x1 c,.5

- *start position within audio file in seconds*

  tie x5 ru,0,10

- *set random panning*

  tie n ru,0,1

- *create a few more instances*

  ticp a b c d e f

- eln; elr; elh

14.14. Polyphonic Sample Grains in athenaCL from a Multiple Audio Files: LineGroove

- Read segments from an audio file by specifying the audio file (with the DirectorySelect PO) and a start time

- Command sequence:

  - emo cn

  - tmo LineGroove

  - *instrument 32 is a fixed playback rate sample player*

    tin a 32

  - *set a file path to an directory, a file extension, and a selection method*

    tie x6 ds,/Volumes/xdisc/_sync/_x/src/martingale/martingale/audio,.aif,rp

  - *set a event time between 60 and 120 ms*

tie r cs,(ru,.060,.120)

- *smooth envelope shapes*

  tie x0 c,.01; tie x1 c,.5

- *start position within audio file in seconds*

  tie x5 ru,0,10

- *set random panning*

  tie n ru,0,1

- *control a variety of amplitudes*

  tie a ru,.2,.4

- *create a few more instances*

  ticp a b c

- eln; elr; elh

14.15. Polyphonic Sample Grains in athenaCL from Multiple Audio Files: TimeFill

- Use TimeFill to create dynamic changes in the density of sampled files
- Command sequence:
  - emo cn
  - tmo TimeFill
  - *instrument 32 is a fixed playback rate sample player*

    tin a 32

  - *set a file path to an directory, a file extension, and a selection method*

    tie x6 ds,/Volumes/xdisc/_sync/_x/src/martingale/martingale/audio,.aif,rp

  - *set a event time between 60 and 120 ms*

    tie r cs,(ru,.030,.090)

  - *smooth envelope shapes*

tie x0 c,.01; tie x1 c,.5

- *start position within audio file in seconds*

  tie x5 ru,0,10

- *set random panning*

  tie n ru,0,1

- *control a variety of amplitudes*

  tie a ru,.1,.2

- *set number of events*

  tie s3 1000

- *start position within texture normalized within unit interval*

  tie d0 rb,.3,.3,0,1

- eln; elr; elh

14.16. Reading: Sturm, Adaptive Concatenative Sound Synthesis

- Sturm, B. L. 2006. "Adaptive Concatenative Sound Synthesis and Its Application to Micromontage Composition." *Computer Music Journal* 30(4): 46-66.

- Sound examples

  http://www.mat.ucsb.edu/~b.sturm/CMJ2006/MATConcat.html

- What are some practical applications of concatenative sound synthesis?

- How is adaptive concatenative sound synthesis a type of analysis and resynthesis, similar to Markov analysis and generation?

- What are some common sonic features used to select source audio?

- Is concatenative sound synthesis algorithmic composition?

Chapter 15. Meeting 15, Approaches: Mapping, Sonification, and Data Bending

15.1. Announcements

- Musical Design Report 3 due 6 April
- Schedule meetings with me over the next two weeks
- Sonic system draft due: 27 April

15.2. Quiz

- 10 Minutes

15.3. Mapping

- Mapping is the conversion of one form or range of data to another
- Most mappings are one to one, and may involve scaling
- Some mappings may be dynamic or procedural
- Sonification refers to mappings of data to musical parameters where perception of the original data is valued
- Data bending refers to mappings of data to musical parameters where the musical output is most valued
- Mappings are often more important than what is being mapped

15.4. Reading: Ben-Tal, O. and J. Berger, Creative Aspects of Sonification

- Ben-Tal, O. and J. Berger. 2004. "Creative Aspects of Sonification." *Leonardo Music Journal* 37(3): 229-232.
- How do the authors describe the difference between musical listening and sonification listening?
- What data sources, and what parameters, do they describe using as source material?
- What arguments support the use of vowel-like synthesized tones? Specifically, how do they parameterize these sounds?

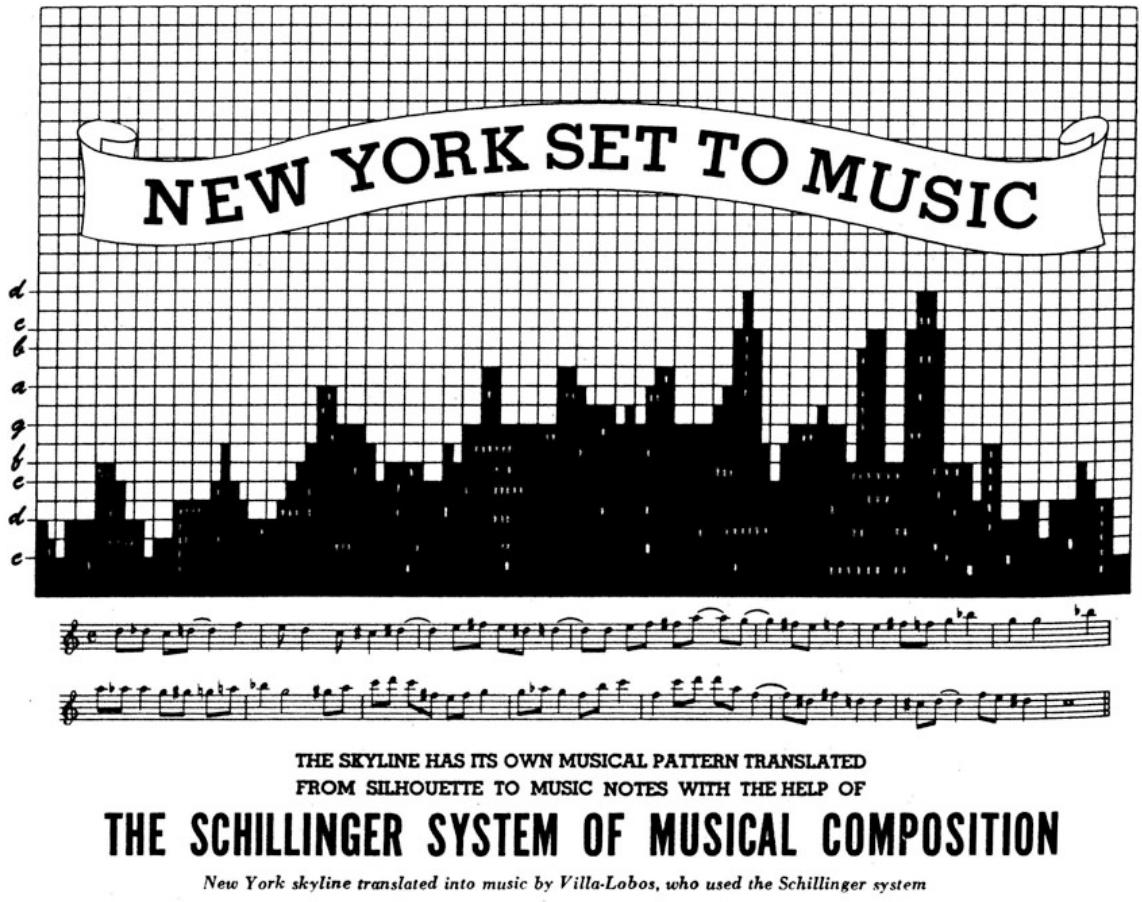
- What advantages do sonification, as type of auditory display, have over visual displays?

15.5. Data Bending

- Mapping of arbitrary data to musical parameters
- Data sonification
- Macro: can be applied to note-level parameters
- Micro: can be applied at the sample level
- Always requires some sort of mapping

15.6. Macro Data Bending: Joseph Schillinger

- Joseph Schillinger (1895-1943): Russian immigrant to the US
- Schillinger, J. 1941. *The Schillinger System of Musical Composition*. New York: Carl Fischer.
- Schillinger, J. 1948. *The Mathematical Basis of the Arts*. New York: Carl Fischer.
- Explored relationships of musical composition to mathematics
- Explored approaches to generating musical parameters



15.7. Macro Data Bending: Natural Data

- Audio: Charles Dodge: "Earth's Magnetic Fields", 1970
- Musical setting of values produced by an index of the effect of the sun's radiation on the magnetic field that surrounds the Earth
- Larry Austin: Canadian Coastlines
- Tracings of outlines of Canadian bodies of water to choose musical parameters such as pitch, rhythm, timbre, and duration

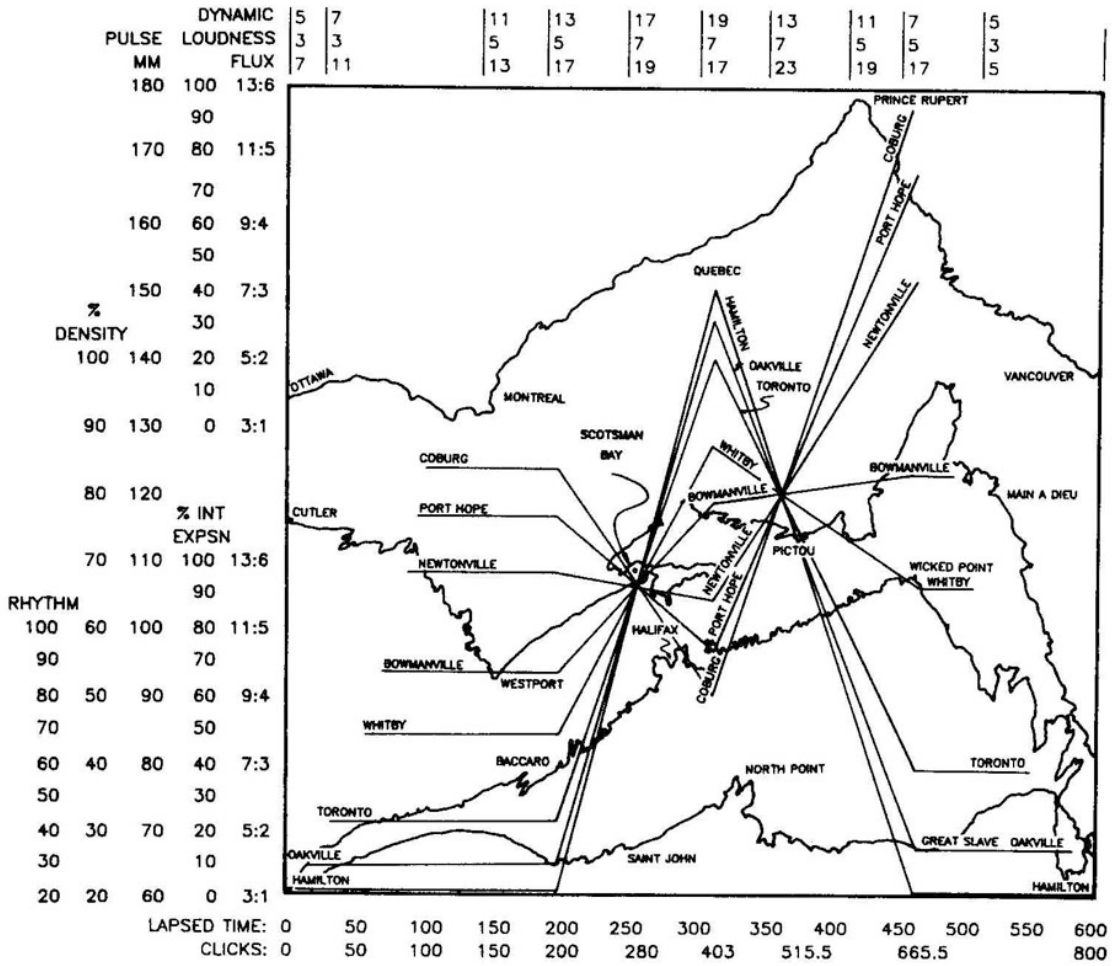


FIGURE 11.21 Chart used by Larry Austin in the composition of *Canadian Coastlines*. *(Reprinted with permission of Larry Austin.)*

Courtesy of Larry Austin. Used with permission.

15.8. Macro Data Bending: Natural Data

- DNA, amino acid sequences in protein, and other natural data sets have frequently been used as a source material for procedural compositions
- Often, claims of aesthetic value are attributed due only to the source data; mapping is incorrectly treated as neutral
- Mary Ann Clark

  Numerous examples created with ArtWonk software from Algorithmic Arts

  http://www.whozoo.org/mac/Music/samples.htm
- Linda Long: Music of the Plants

http://www.molecularmusic.com

15.9. Micro Data Bending

- Mapping non-musical data to audio-rate data
- Can map to an arbitrary binary data representation
- Can map to positions of individual amplitude points

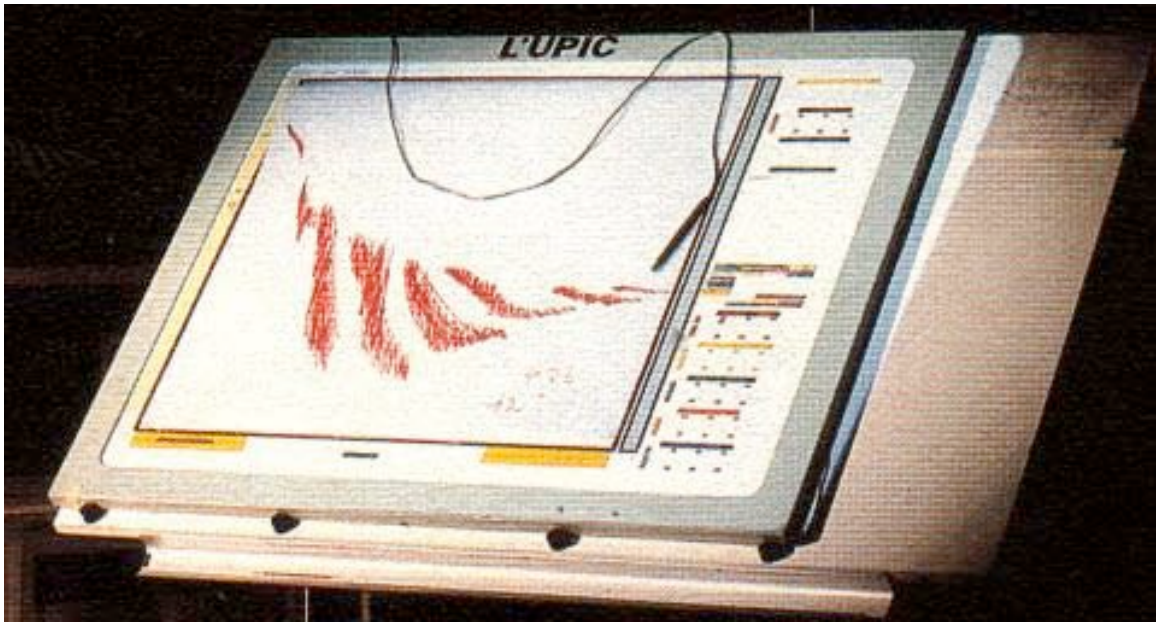
15.10. Micro Data Bending: Arbitrary Data as Audio File

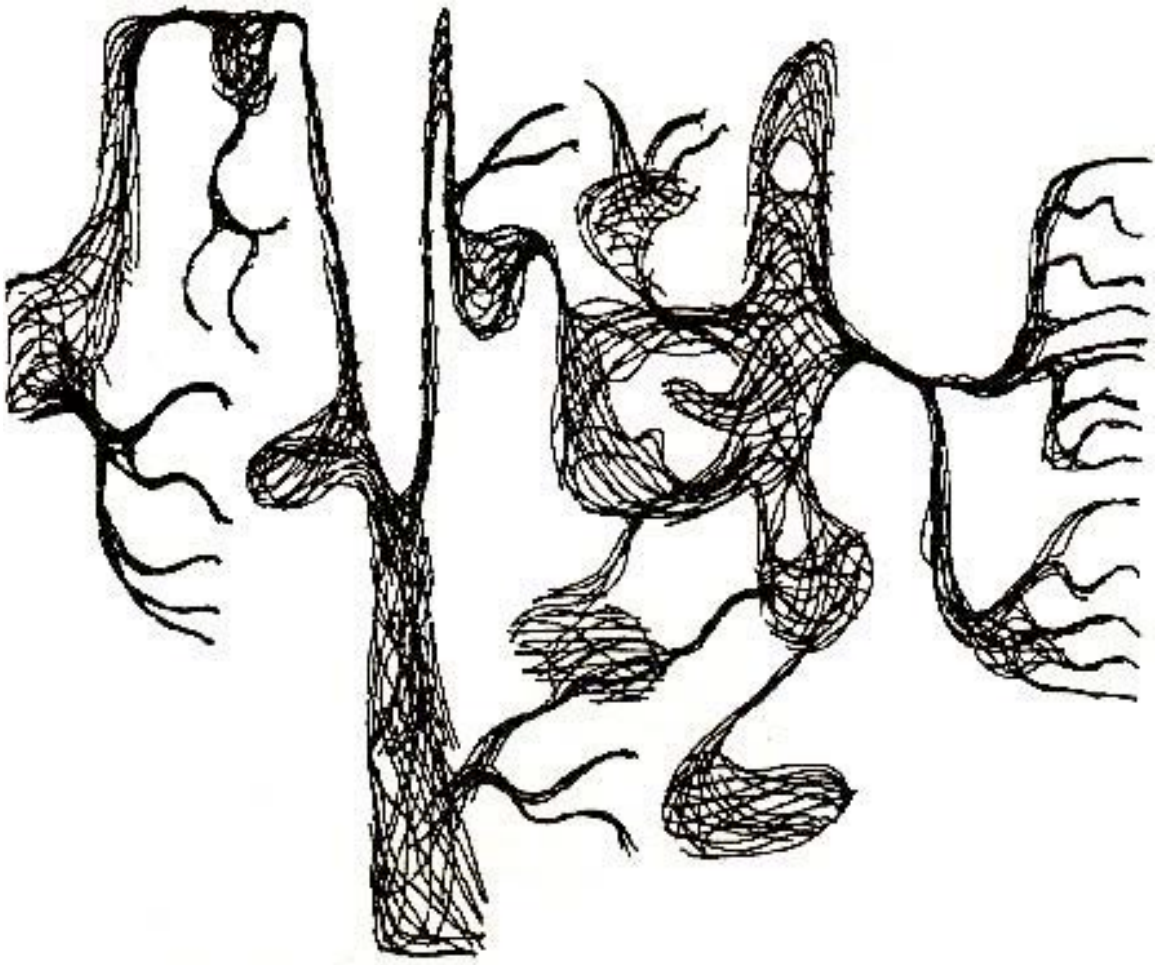
- Read images other file types as audio data: UPIC, MetaSynth
- Import arbitrary data as audio data
- Audacity: Project: Import Raw Data...

  Example: AN0008-amesRetrospect.pdf

15.11. UPIC

- Unité Polyagogique Informatique du CEMAMu (UPIC),
- Users draw waveforms, envelopes, and textures
- Integrated hardware system





- Audio: "Mycenae-Alpha," 1978
- Mycenae-Alpha visualization synchronized to the score

  Video: YouTube (http://www.youtube.com/watch?v=yztoaNakKok)
- "Children may draw a fish or a house and listen to what they have made and correct it. They can learn, progressively through designing, to think musical composition without being tormented by solfège or by incomplete mastery of a musical instrument.... But as they are led to construct rhythms, scales, and more complex things, they are also forced to combine arithmetic and geometric forms: music. From whence comes an interdisciplinary pedagogy through playing." (Xenakis 1985).

15.12. Reading: Marino, G. and M. Serra, J. Raczinski, The UPIC System: Origins and Innovations

- Marino, G. and M. Serra, J. Raczinski. 1993. "The UPIC System: Origins and Innovations." *Perspectives of New Music* 31(1): 258-269.

- Evaluate this idea and claim: "Another idea was to let the composer control and create all aspects of the composition: sound, symbols, syntax, and so forth. This means that the system should not impose predefined sounds, predefined compositional process, predefined structures, and so on. It is essential for the creative mind that ideas not go through theories or limitations that might not suit the composer."

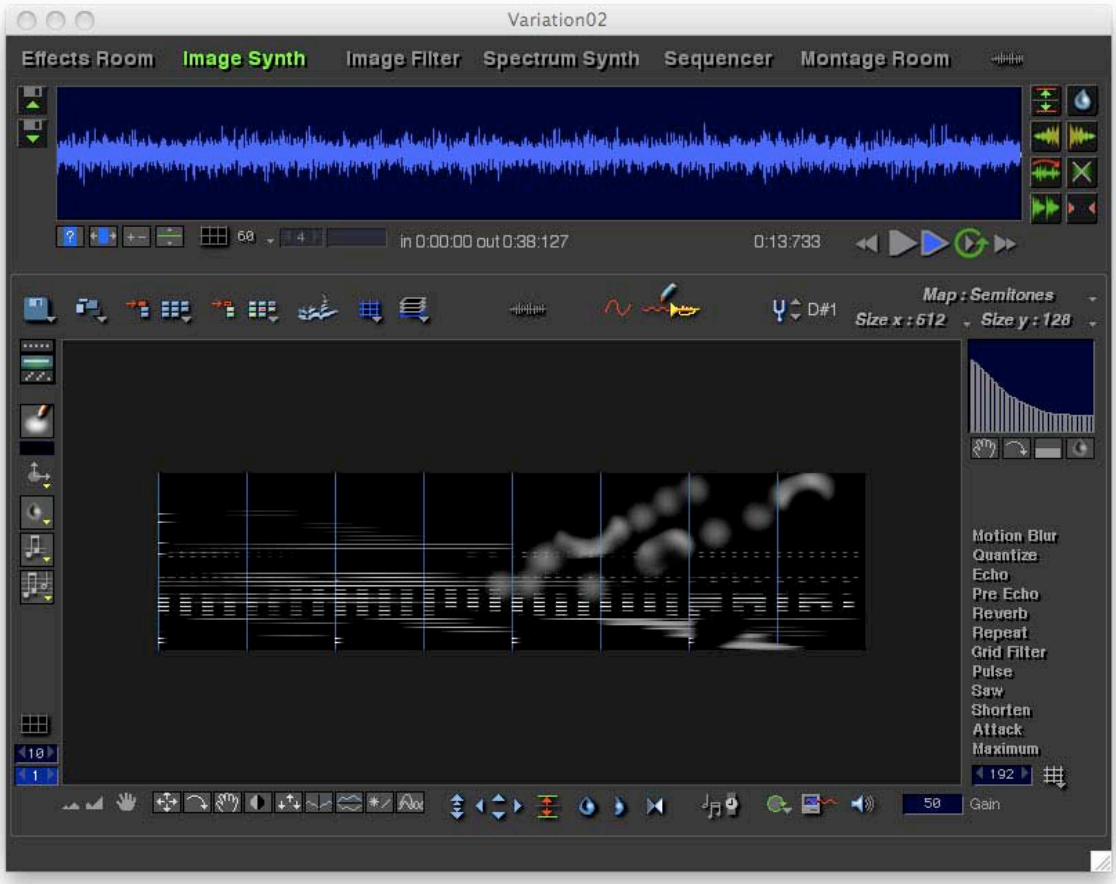
- What are some of the technical features of this version of UPIC?

- Where (on what machine components) does the signal processing occur?

- What opportunities exist for non-sinusoidal sounds?

15.13. Contemporary UPIC Variants

- MetaSynth (Mac; commercial)



http://www.uisoftware.com/MetaSynth

- HighC: (Windows, Mac, Unix; commercial)

  http://highc.org

- SPEAR (Windows, Mac; free)

  Designed for audio analysis and resynthesis; permits drawing spectra

  http://www.klingbeil.com/spear/

- HyperScore: Anyone Can Compose Music

  A note-event (not synthesis) approach to writing common practice (tonal, pitched) music

  http://www.hyperscore.com

Chapter 16. Meeting 16, Workshop

16.1. Announcements

- Musical Design Report 3 due Today
- Schedule meetings with me over this and next week
- Sonic system draft due: 27 April
- Next Quiz: Thursday, 15 April (inclusive)

16.2. Quiz Review

- ?

16.3. Workshop: Musical Design Report 3

- Three students give their reports today

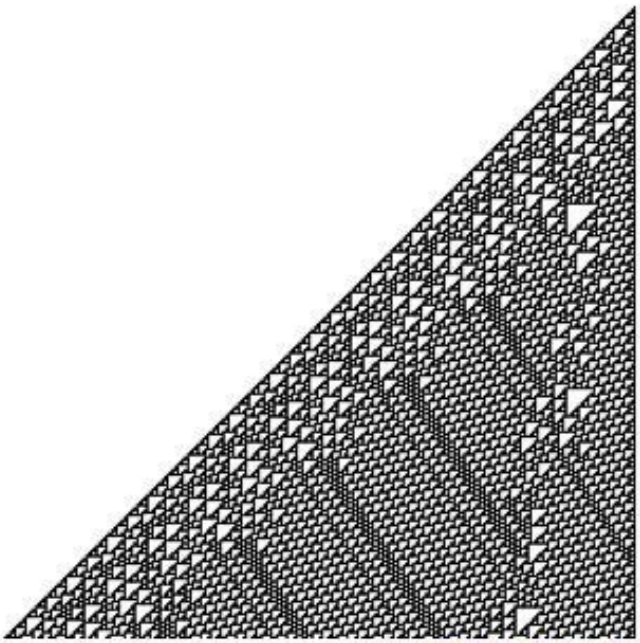
Chapter 17. Meeting 17, Approaches: Cellular Automata

17.1. Announcements

- Schedule meetings with me over this week
- Sonic system draft due: 27 April
- Next Quiz: Thursday, 15 April (inclusive)

17.2. Cellular Automata

- The iterative application of a rule on a set of states
- States are organized in a lattice of cells in one or more dimensions
- To determine the *n* state of the lattice, apply a rule that maps *n-1* to *n* based on contiguous sections of cells (a neighborhood)
- A rule set contains numerous individual rules for each neighborhood



Rule 110 cellular automaton

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

- CA are commonly described as having four types of behavior (after Wolfram): stable homogeneous, oscillating or patterned, chaotic, complex

17.3. CA History

- 1966: John von Neimann demonstrates a 2D, 29-state CA capable of universal computation
- 1971: Edwin Roger Bank demonstrates 2D binary state CA
- 2004: Matthew Cook demonstrates 1D binary state, rule 110 CA

17.4. CA in Music

- First published studies: Chareyron (1988, 1990) and Beyls (1989)
- Chareyron: applied CA to waveforms
- Beyls: numerous studies applied to conventional parameters
- Xenakis: employed CA in Horos (1986)

Mapped CA to a large scale and used active cells to select pitches

17.5. The caSpec

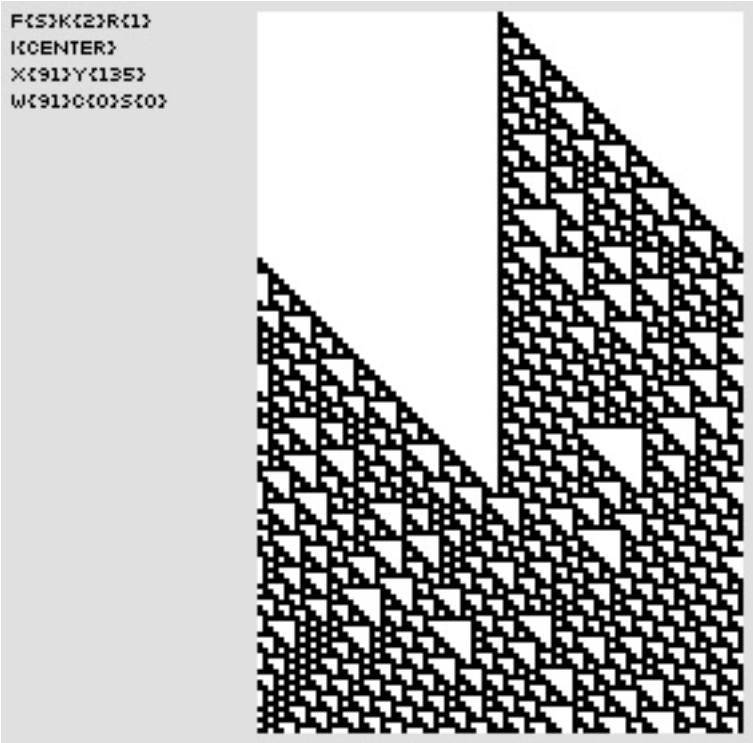
- String-based notation of CA forms
- Key-value pairs: key{value}

17.6. CA Types

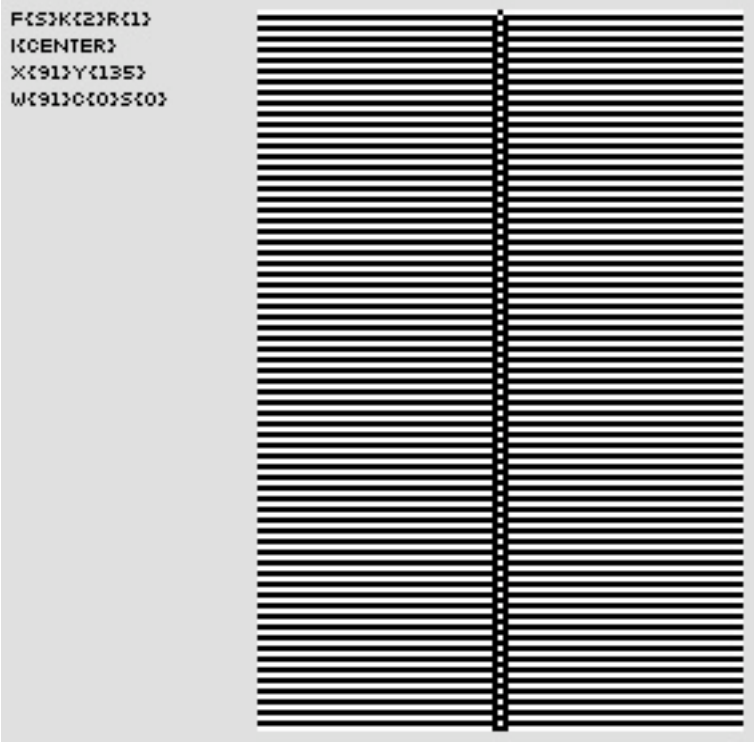
- Standard: f{s}

  Discrete cell values, rules match cell formations (neighborhoods)

```
:: auca f{s} 380 0
f{s}k{2}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



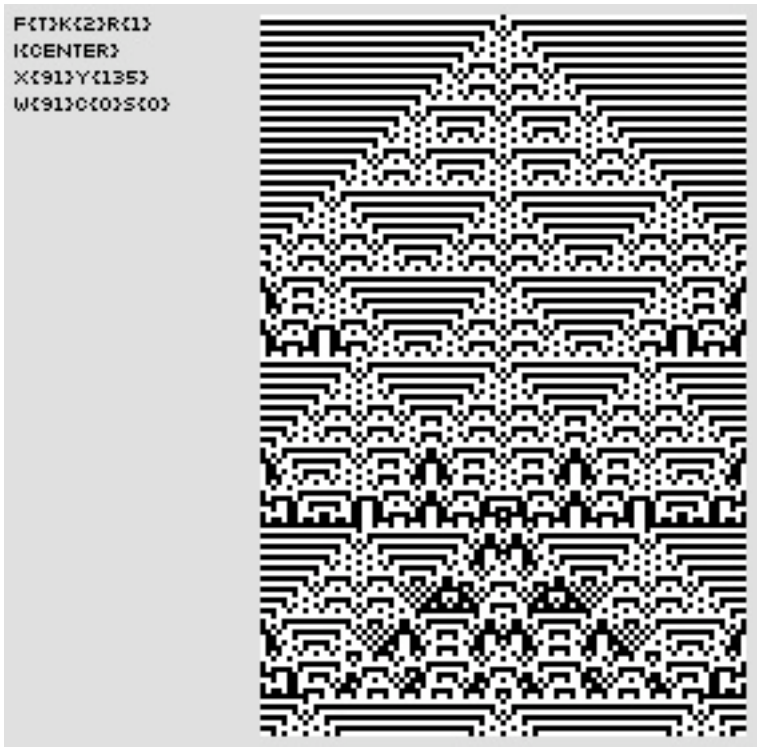

```
:: auca f{s} 379 0
f{s}k{2}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



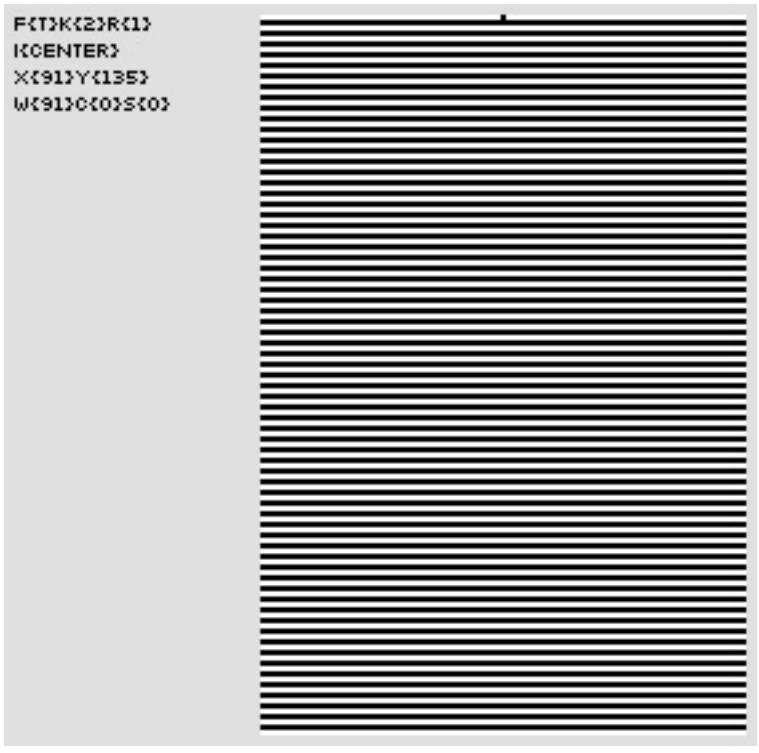
- Totalistic: f{t}

  Discrete cell values, rules match the sum of the neighborhood

```
:: auca f{t} 37 0
f{t}k{2}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



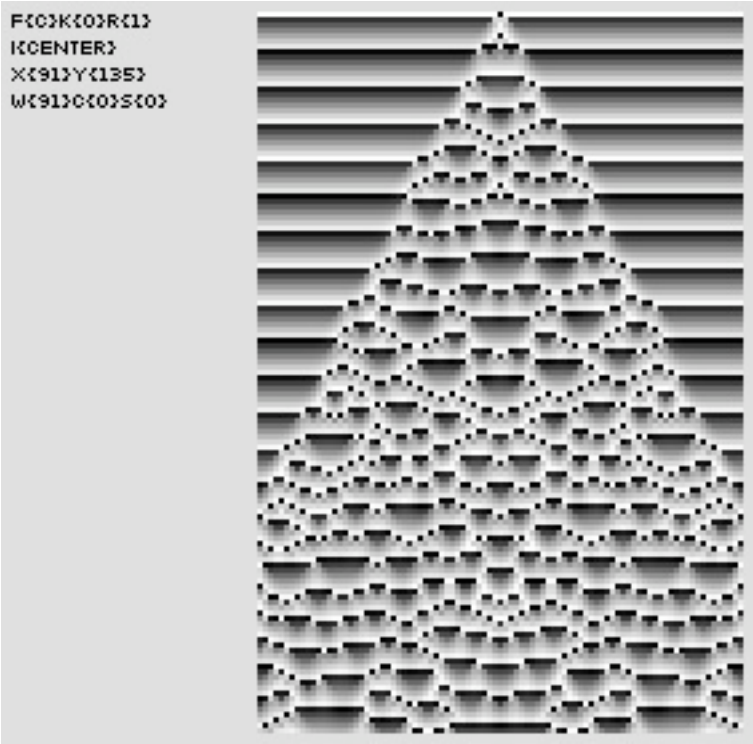

```
:: auca f{t} 39 0
f{t}k{2}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



- Continuous: f{c}

  Real-number cell values within unit interval, rules specify values added to the average of previous cell formation

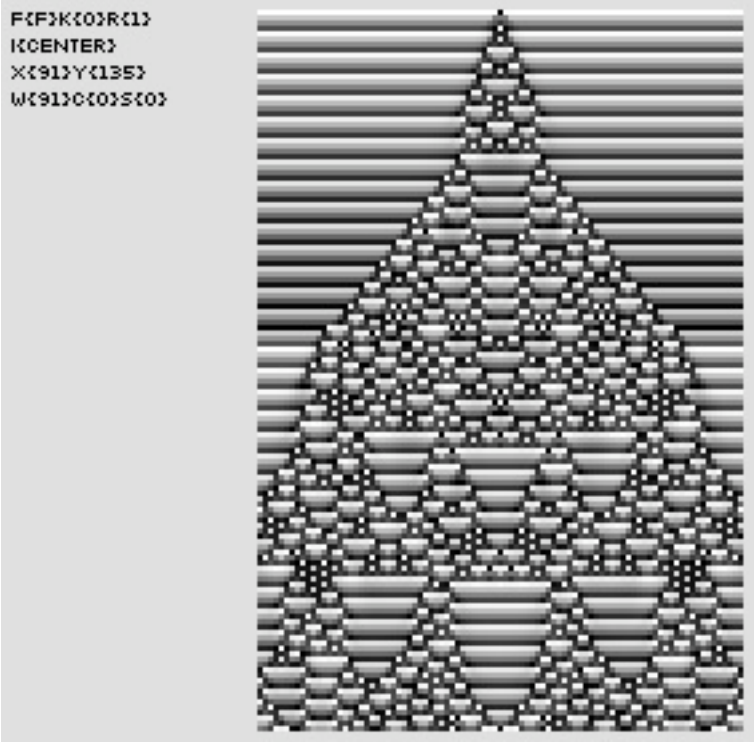
```
:: auca f{c} .8523 0
f{c}k{0}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



- Float: f{f}

  Like continuous, but implemented with floats (it makes a difference)

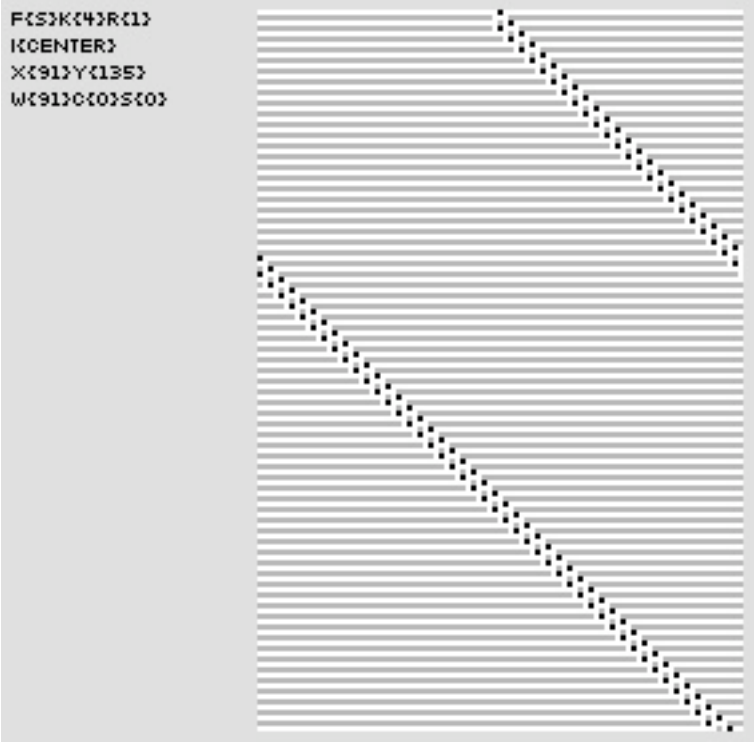
```
:: auca f{f} .254 0
f{f}k{0}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



17.7. Possible Cell States

- For f{s,t}: the *k* value provides the number of possible values
- For f{c,f}: the *k* value is zero
- The k value can be set for discrete CA

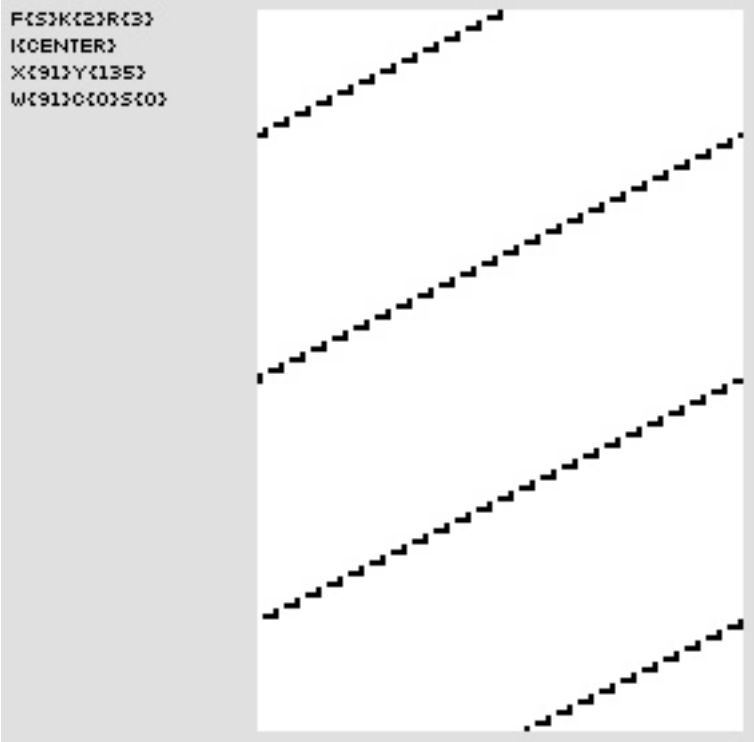
```
:: auca f{s}k{4} 3841 0
f{s}k{4}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



17.8. Rules Neighorhood

- The *r* number defines the number of cell states taken into account
- For 1D CA, the neighborhood is 2*r*+1
- Half integer fractional values are permitted
- An r{3} CA

```
:: auca f{s}r{3} 380 0
f{s}k{2}r{3}i{center}x{91}y{135}w{91}c{0}s{0}
```



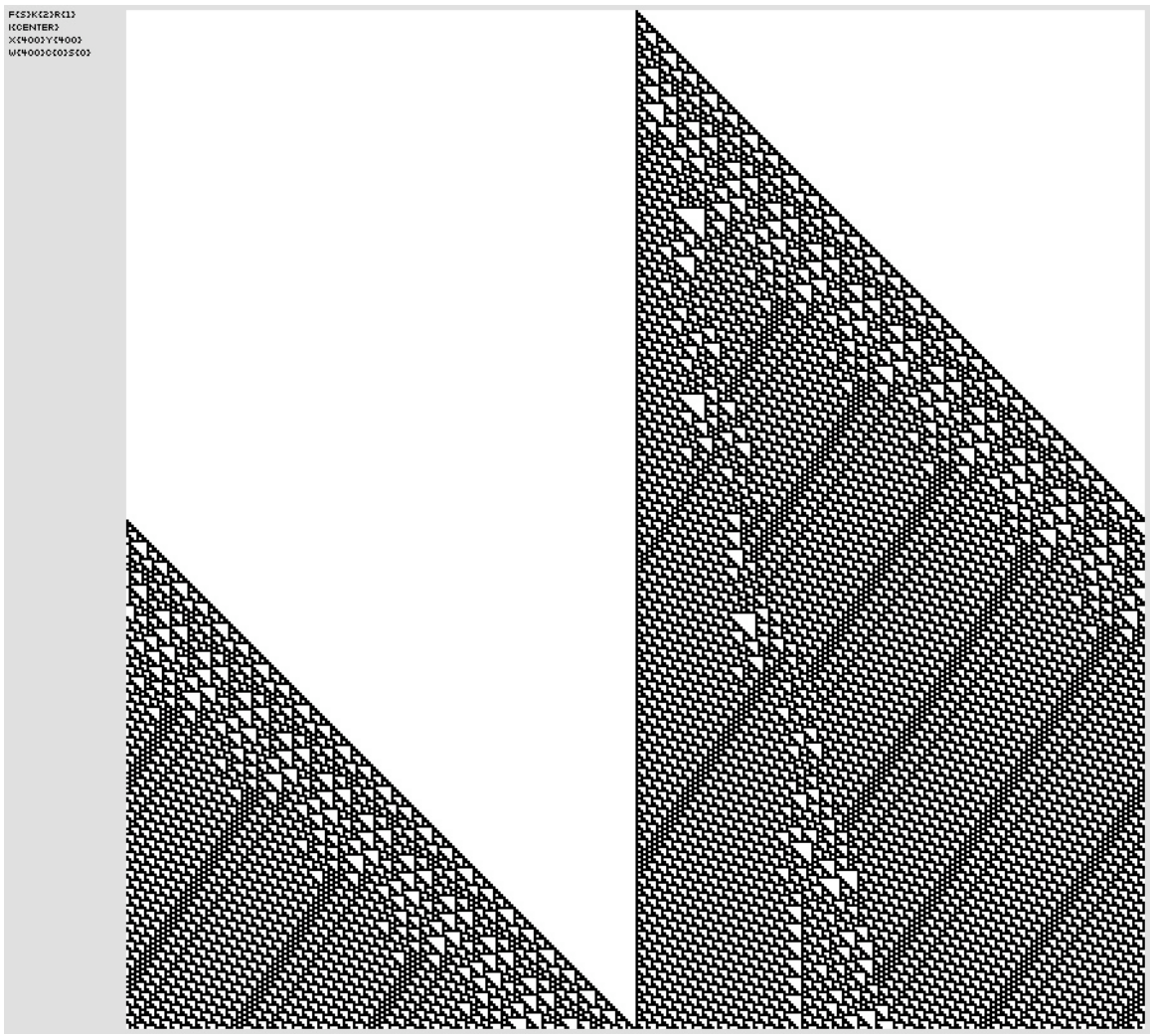
17.9. Size, Orientation, and Presentation

- 1D often present 1 horizontal row that wraps, unbound but finite space
- A table, with cell sites on *x* axis, time on *y* values
- A cylinder
- Size is given with *x*, number of evolutions specified with *y*

```
:: auca f{s}x{9}y{200} 380 0
f{s}k{2}r{1}i{center}x{9}y{200}w{9}c{0}s{0}
```

F{S}K{2}R{1}
I{CENTER}
X{9}Y{200}
W{9}C{0}S{0}

```
:: auca f{s}x{400}y{400} 380 0
f{s}k{2}r{1}i{center}x{400}y{400}w{400}c{0}s{0}
```



- Can specify a sub-table with a width and a center independent of *x* axis, time on *y* values

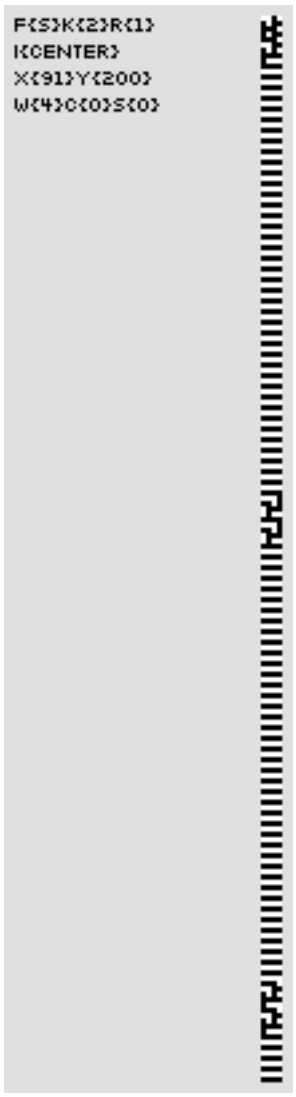
  Width, w{}, is the number of exposed cells

  Center, c{}, is center position from which cells are extracted

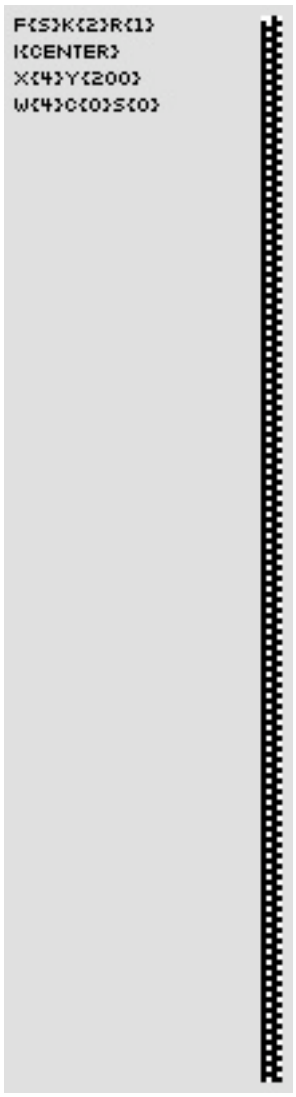
  Skip, s{}, is the number of rows neither displayed nor counter in *y*.

- Example: a width is not the same as

```
:: auca f{s}x{91}y{200}w{4} 381 0
f{s}k{2}r{1}i{center}x{91}y{200}w{4}c{0}s{0}
```



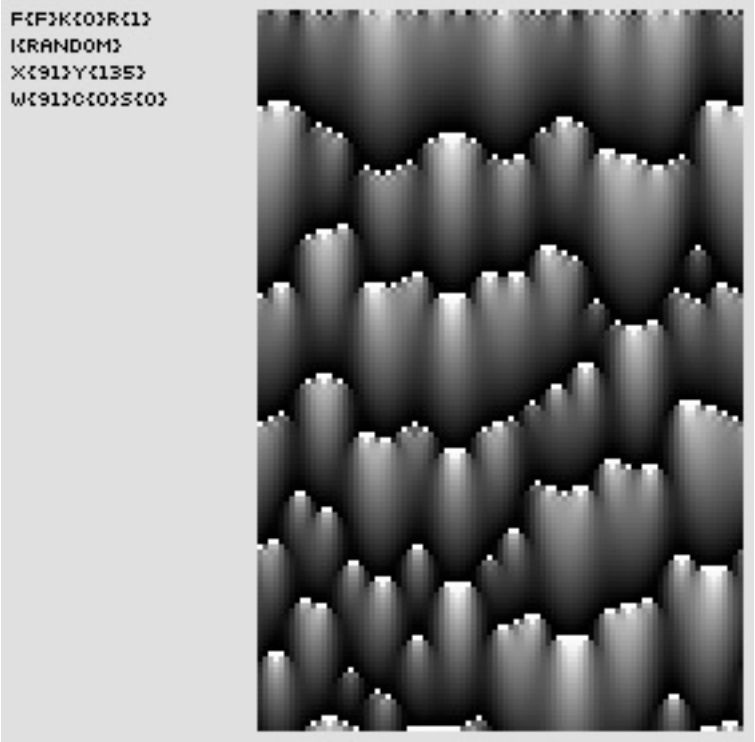

```
:: auca f{s}x{4}y{200} 381 0
f{s}k{2}r{1}i{center}x{4}y{200}w{4}c{0}s{0}
```



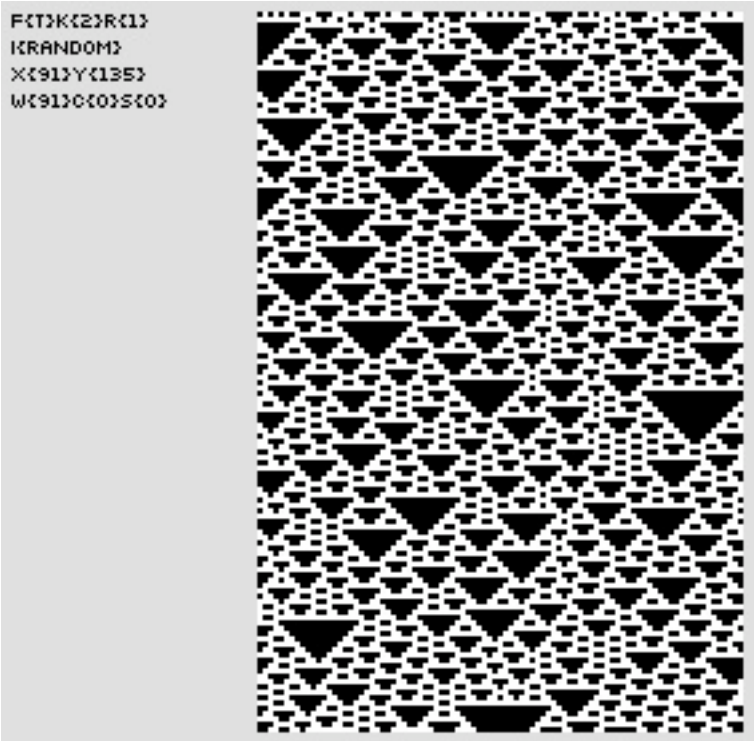
17.10. The Initial Row

- The init can be specified with an i{} parameter
- Strings like center (i{c}) and random (i{r}) are permitted

```
:: auca f{f}i{r} .0201 0
f{f}k{0}r{1}i{random}x{91}y{135}w{91}c{0}s{0}
```

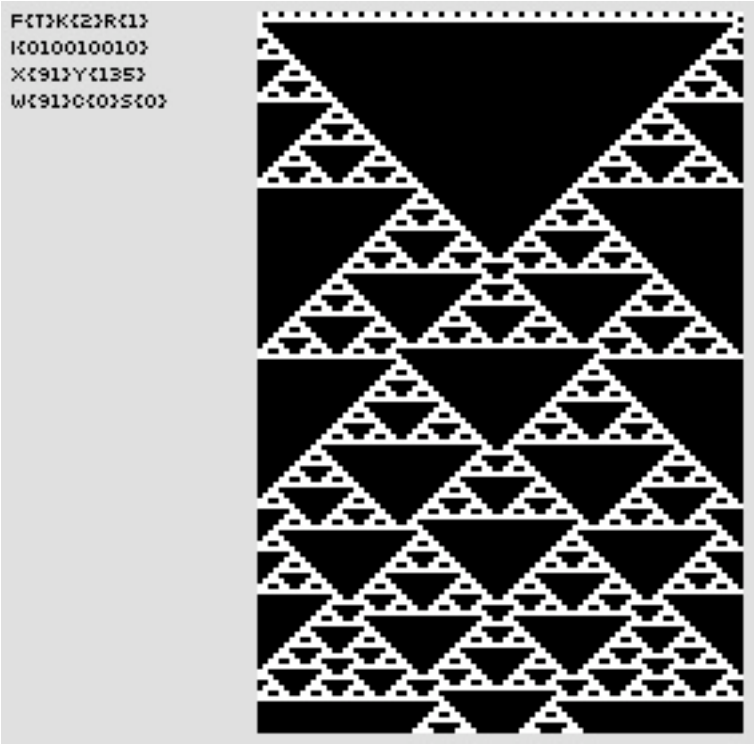



```
:: auca f{t}i{r} 201 0
f{t}k{2}r{1}i{random}x{91}y{135}w{91}c{0}s{0}
```



- Numerical sequences of initial values repeated across a row

```
:: auca f{t}i{010010010} 201 0
f{t}k{2}r{1}i{010010010}x{91}y{135}w{91}c{0}s{0}
```



17.11. Dynamic Parameters: Rule and Mutation

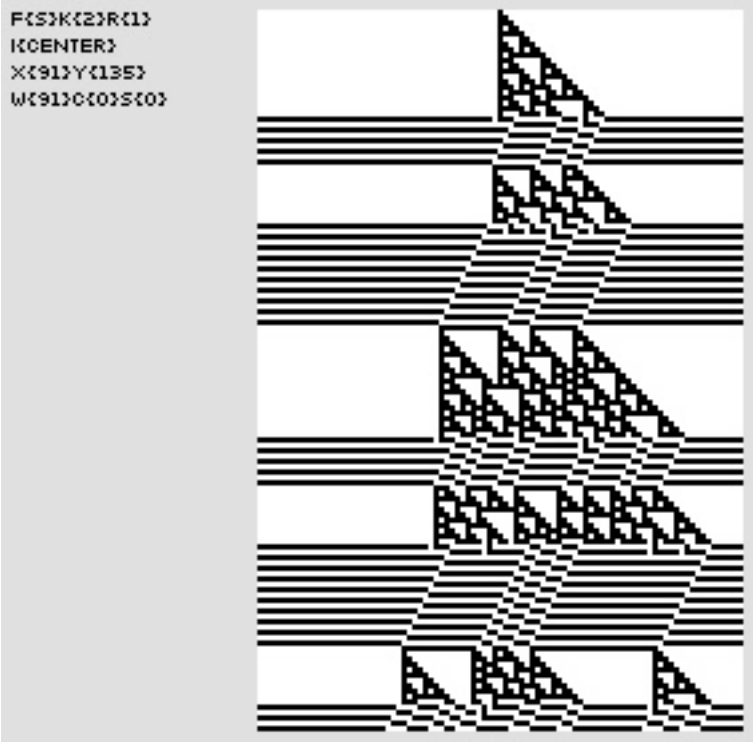
- Rule: a floating or integer value

  Wolfram offers standard encoding of rules as integers

  Out of range rule values are resolved by modulus of total number of rules

- PO applied to the rule value of CA

```
:: auca f{s} ig,(bg,rp,(380,533)),(bg,rp,(10,20)) 0
f{s}k{2}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



- Mutation: a unit interval probability
- PO applied to the mutation of a CA

```
:: auca f{s} 533 whpt,e,(bg,rp,(8,16,32,64)),0,.01
f{s}k{2}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```

17.12. Reading: Ariza: Automata Bending: Applications of Dynamic Mutation and Dynamic Rules in Modular One-Dimensional Cellular Automata

- Ariza, C. 2007a. "Automata Bending: Applications of Dynamic Mutation and Dynamic Rules in Modular One-Dimensional Cellular Automata." *Computer Music Journal* 31(1): 29-49. Internet: http://www.mitpressjournals.org/doi/abs/10.1162/comj.2007.31.1.29.
- What is automata bending? Why has this not been previously explored?
- What are the benefits of automata bending for creative applications?
- "The utility and diversity of CA are frequently overstated": is this statement warranted?
- What are some of the problems of using CA that do exhibit emergent
- What does Wolfram think of float CA. Is he right?
- Hoffman claims that Xenakis's use of CA demonstrated "the strength and limitation of universal computation in music composition"; is this possible?

17.13. Bent Automata

- Examples

```
:: auca f{s}x{81}y{80}k{2}r{1} 109 0.003

:: auca f{t}x{81}y{80}k{3}r{1} 1842 bpl,e,l,((0,0),(80,.02))

:: auca f{s}x{81}y{80}k{2}r{1}i{r} 90.5 0

:: auca f{t}y{80}x{81}r{1}k{4}i{r}s{20} mv,a{195735784}b{846484}:{a=3|b=1} 0
```

17.14. Mapping Tables to Single Value Data Streams

- Combinations of type, axis, source, filter, 60 total possibilities

Table 1. Table Extraction Parameters (an Asterisk Designates a Default Parameter)

Parameters				*Methods*	
Type	*Axis*	*Source*	*Filter*	*Count*	*Examples*
flat	row column rowReflect columnReflect	value* index passive	none* active	24	flatRowActive flatRowReflectIndexActive flatColumnIndex flatColumnReflectPassive
sum product average	row column	value* index	none* active passive	36	sumRow productColumnIndexPassive averageRowActive

Source: Ariza, C. *Computer Music Journal* 31, no. 1 (2007): 29-49.

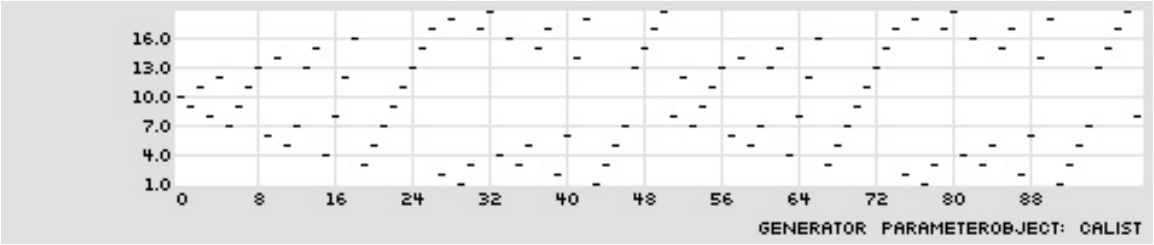
17.15. The CA as ParameterObject

- All underlying tools for automata are found in automata.py
- CaList and CaValue provide high level ParameterObject interfaces
- CaList returns raw CA values (processed by table extraction) that can be selected from using common selectors; CaValue normalizes within unit interval and provides dynamic min and max values

17.16. The CA as a Generator of Melodies

- Probably the most common approach: use active cell index positions to indicate active positions of a scale
- CaList with rule 90 and flatRowIndexActive; a smaller x is used to reduce index values

```
:: tpmap 100 cl,f{s}x{20},90,0,fria,oc
caList, f{s}k{2}r{1}i{center}x{20}y{135}w{20}c{0}s{0}, (constant, 90),
(constant, 0), flatRowIndexActive, orderedCyclic
TPmap display complete.
```



- Command sequence using TM Harmonic Assembly:
  - emo m
  - *create a single, large Multiset using a sieve*

    pin a 5@0|7@2,c2,c7
  - tmo ha
  - tin a 27
  - tie r pt,(c,8),(ig,(bg,rc,(2,3)),(bg,rc,(3,6,9))),(c,1)
  - tie a ls,e,9,(ru,.2,1),(ru,.2,1)
  - *select only Multiset 0*

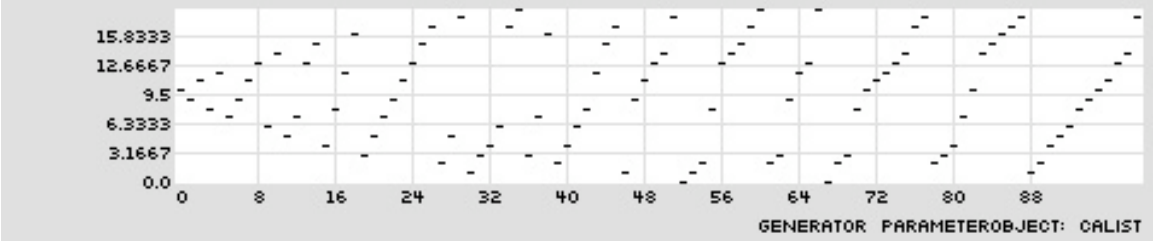
    tie d0 c,0
  - *select pitches from Multiset using CaList*

    tie d1 cl,f{s}x{20},90,0,fria,oc
  - *create only 1 simultaneity from each multiset*

    tie d2 c,1
  - *create only 1-element simultaneities*

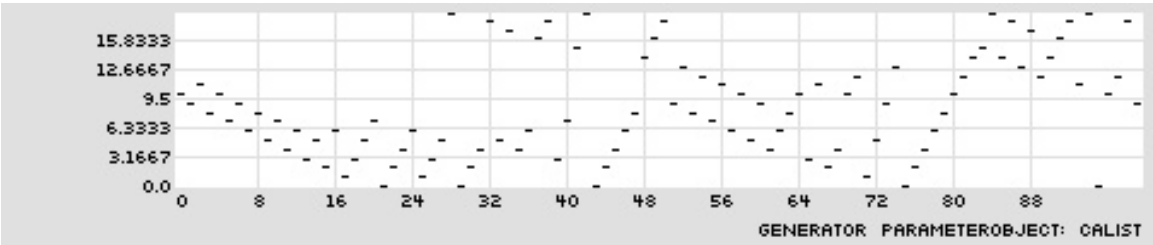
    tie d3 c,1
  - eln; elh
- CaList with rule 90 and flatRowIndexActive; a smaller x is used to reduce index values; adding mutation

```
:: tpmap 100 cl,f{s}x{20},90,(ls,e,16,0,.05),fria,oc
caList, f{s}k{2}r{1}i{center}x{20}y{135}w{20}c{0}s{0}, (constant, 90), (lineSegment,
(constant, 16), (constant, 0), (constant, 0.05)),
flatRowIndexActive, orderedCyclic
TPmap display complete.
```



- CaList with a mixture of rule 90 and rule 42 and flatRowIndexActive; a smaller x is used to reduce index values; adding mutation

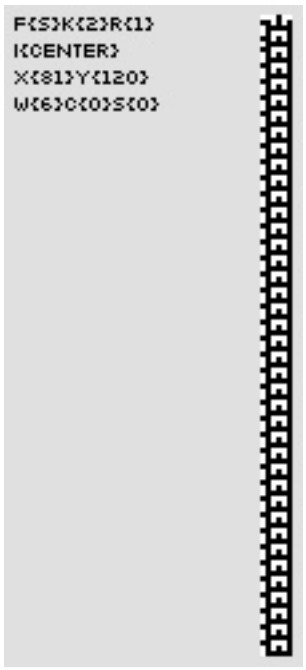
```
:: tpmap 100 cl,f{s}x{20},(ig,(bg,rp,(90,42)),(bg,rp,(2,3))),0,fria,oc
caList, f{s}k{2}r{1}i{center}x{20}y{135}w{20}c{0}s{0}, (iterateGroup, (basketGen,
randomPermutate, (90,42)), (basketGen, randomPermutate, (2,3))),
(constant, 0), flatRowIndexActive, orderedCyclic
TPmap display complete.
```



17.17. The CA as a Generator of Rhythms

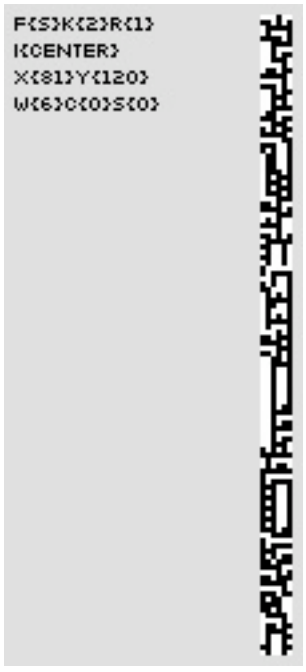
- Narrow regions of bent CA offer interesting variation of few values
- A a narrow width of a CA

```
:: auca f{s}k{2}r{1}x{81}y{120}w{6}c{0}s{0} 109 0
f{s}k{2}r{1}i{center}x{81}y{120}w{6}c{0}s{0}
```



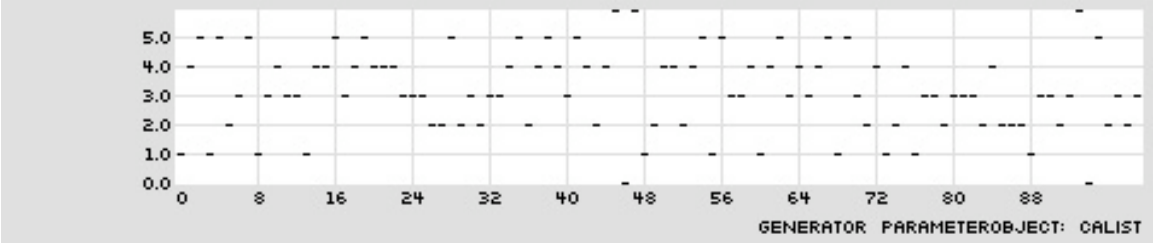
- A a narrow width of a CA with a small constant mutation

```
:: auca f{s}k{2}r{1}x{81}y{120}w{6}c{0}s{0} 109 .05
f{s}k{2}r{1}i{center}x{81}y{120}w{6}c{0}s{0}
```



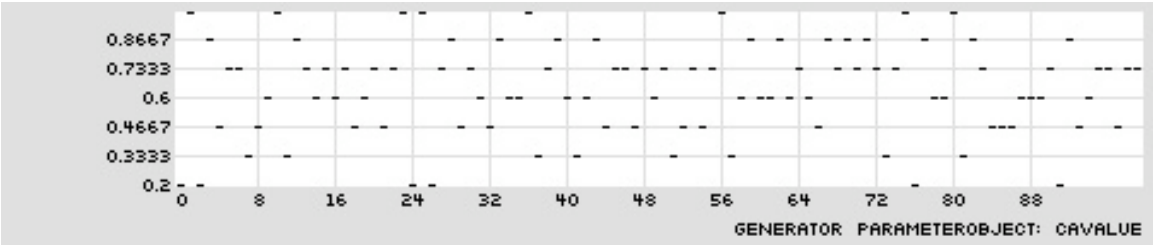
- Using CaTable and sumRowActive, we can get a dynamic collection of small integer values

```
:: tpmap 100 cl,f{s}k{2}r{1}x{81}y{120}w{6}c{0}s{0},109,.05,sumRowActive,oc
caList, f{s}k{2}r{1}i{center}x{81}y{120}w{6}c{0}s{0}, (constant, 109), (constant,
0.05), sumRowActive, orderedCyclic
TPmap display complete.
```



- Using CaValue and sumRowActive with a different center, we can get a dynamic collection of floating point values

```
:: tpmap 100 cv,f{s}k{2}r{1}x{81}y{120}w{6}c{8}s{0},109,.05,sumRowActive,.2,1
caValue, f{s}k{2}r{1}i{center}x{81}y{120}w{6}c{8}s{0}, (constant, 109), (constant,
0.05), sumRowActive, (constant, 0.2), (constant, 1),
orderedCyclic
TPmap display complete.
```



- Command sequence using TM Harmonic Assembly:
  - emo mp
  - tin a 47
  - *set the multiplier to the integer output of CaList*

    tie r pt,(c,4),(cl,f{s}k{2}r{1}x{81}y{120}w{6}c{0}s{0},109,.05,sumRowActive,oc),(c,1)
  - *set the amplitude to the floating potin output of CaValue*

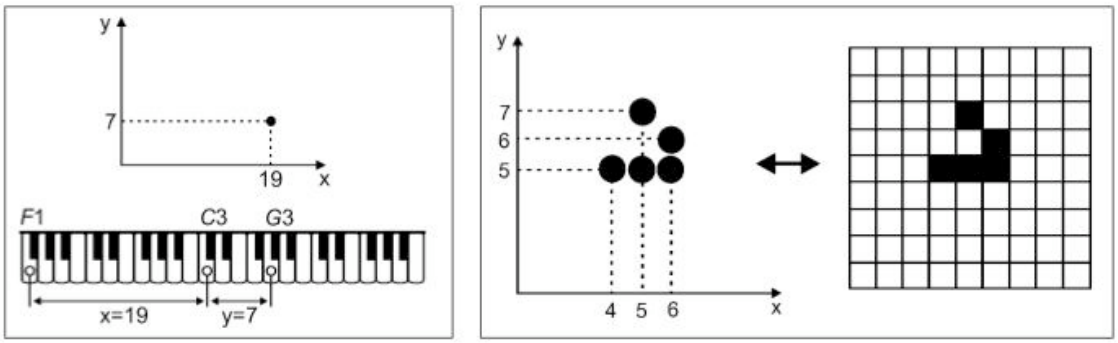
    tie a cv,f{s}k{2}r{1}x{81}y{120}w{6}c{8}s{0},109,.05,sumRowActive,.2,1
  - eln; elh

17.18. Reading: Miranda: On the Music of Emergent Behavior: What Can Evolutionary Computation Bring to the Musician?

- Miranda, E. R. 2003. "On the Music of Emergent Behavior: What Can Evolutionary Computation Bring to the Musician?." *Leonardo* 36(1): 55-59.

- Miranda claims that "the computer should neither be embedded with particular models at the outset nor learn from carefully selected examples"; is this possible, and is this achieved with his model?

- What is the basic mapping of CAMUS?



Courtesy of MIT Press. Used with permission.

- What is the basic mapping of Chaosynth?

- What does Miranda mean when he states that "none of the pieces cited above were entirely automatically generated by the computer"; is this possible?

Chapter 18. Meeting 18, Approaches: Genetic Algorithms

18.1. Announcements

- Next Quiz: Thursday, 15 April (inclusive)
- Sonic system draft due: 27 April
- No class Tuesday, 20 April

18.2. Genetic Algorithms

- Model states of a system (or processes) as an allele, or a fundamental unit of expression
- Two or more alleles form a chromosome; order of alleles generally is significant
- Chromosomes, representing individuals, are collected in a population
- Using a fitness function, each chromosome is given a fitness value
- Chromosomes are mated under conditions where more-fit chromosomes are more likely to mate
    - Two chromosomes can produce two offspring (replacing themselves)
    - Each new chromosome is created by either cloning parents or intermingling their alleles through one or two-point crossover
    - Each child chromosome may undergo mutation at the level of single allele changes or multiple allele changes
- The population is completely replaced through mating
- Numerous cycles of regeneration are completed
- The goal is for the population to evolve the most fit chromosome

18.3. GA History and Common Applications

- First described in depth by John Holland in 1975

  Holland, J. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Michigan: The University of Michigan Press.

- Employed in tasks ranging from computational protein engineering, automatic programming, and the modeling of economic and ecological systems.

- Generally best suited for solving problems that lack rigorous definition

18.4. Encoding the Alleles and Chromosomes

- Many GA designs use binary encoding: 1s and 0s encode desired parameters
- Real-value encoding uses an alphabet of many characters or real numbers
- Many GAs use fixed length chromosomes

18.5. Mutations

- Binary GAs often perform bit-level manipulations
  - Bits can be flipped
  - Segments of bits can be deleted, repeated, or reversed
- Domain-specific GA mutations are possible

18.6. The Fitness Function and Finding Solutions

- The fitness function is the key
- The fitness function expresses the priorities of the system
- GAs can evolve toward a local fitness yet get stuck, not reaching the maximum fitness
- Some systems have employed human-mediated fitness evaluation

18.7. A GA of Pulse Triple Chromosomes

- Project conducted in 2001-2002

  Ariza, C. 2002. "Prokaryotic Groove: Rhythmic Cycles as Real-Value Encoded Genetic Algorithms." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 561-567. Internet: http://www.flexatone.net/docs/pgrcrvega.pdf.

- First design for sub-system models in athenaCL, exposed through ParameterObjects
- Alleles are pulse triples
- Chromosome is a sequence of alleles where order is musically performed order
- Fitness function is based on similarity to a target chromosome

    - Find temporal distance of note durations, rest durations, and total duration (larger values mean greater distance)
    - Find weighted duration of non-matched alleles (non-exact pulse triple matches, where count is multiplied by average allele duration)
    - Find weighted duration of non-matched duration ratios (non matching pulse triple ratios, where count is multiplied by average allele duration)
    - Sum of these values weighted with values found through experiment: noteDistance*1.50, restDistance*1.50, durDistance*2.33, noMatchAlleleDistance*1.00, noMatchValueDistance*0.66.
    - An inverse relation: the larger the value, the greater the distance from the target
- Two point crossover employed in mating
- Mutations are specific to pulse triples
    - Ratio equivalence: multiply or divide divisor or multiplier by 2 or 3
    - Divisor mutate: add or subtract 1 to divisor
    - Multiplier mutate: add or subtract 1 to multiplier
    - Flip note/rest state
    - Inversion: select to lic, reverse the segment with the retrograde of the segment
- Population is initialized through random arrangements of pulse triples found in the source
- For each generation, retain the chromosome that is the best fit (and is unique)
- After generations are complete, order best-fit chromosomes by fitness
- Example: python genetic.py

18.8. GA as ParameterObject

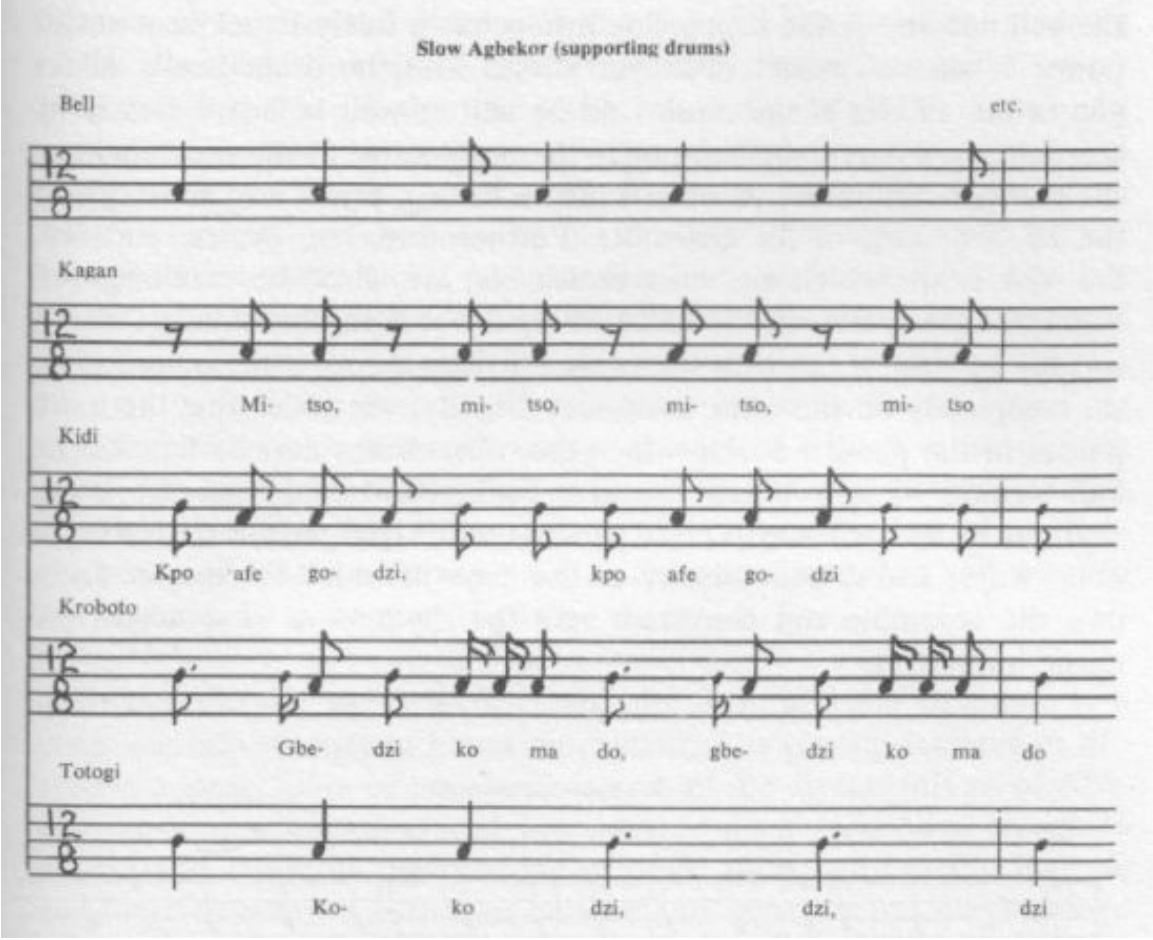
- The gaRhythm ParameterObject

```
:: tpv garhythm
Rhythm Generator ParameterObject
{name,documentation}
GaRhythm            gaRhythm, pulseList, crossover, mutation, elitism,
                    selectionString, populationSize
                    Description: Uses a genetic algorithm to create rhythmic
                    variants of a source rhythm. Crossover rate is a percentage,
                    expressed within the unit interval, of genetic crossings
                    that undergo crossover. Mutation rate is a percentage,
                    expressed within the unit interval, of genetic crossings
```

```
that undergo mutation. Elitism rate is a percentage,
expressed within the unit interval, of the entire population
that passes into the next population unchanged. All rhythms
in the final population are added to a list. Pulses are
chosen from this list using the selector specified by the
control argument. Arguments: (1) name, (2) pulseList {a list
of Pulse notations}, (3) crossover, (4) mutation, (5)
elitism, (6) selectionString {"randomChoice", "randomWalk",
"randomPermutate", "orderedCyclic",
"orderedCyclicRetrograde", "orderedOscillate"}, (7)
populationSize
```

18.9. Evolving African Drum Patterns with a GA

- Slow Agbekor (Chernoff 1979)



- Command sequence 1: exploring two durations:
  - emo mp
  - tmo lg

- tin a 61
- *bell line, set to loop*

  tie r l,[(4,4,1),(4,4,1),(4,2,1),(4,4,1),(4,4,1),(4,4,1),(4,2,1)]
- *accent the first of each articulation*

  tie a bg,oc,(1,.5,.5,.5,.5,.5,.5)
- tin b 68
- *create genetic variations using a high mutation rate*

  tie r gr,[(4,4,1),(4,4,1),(4,2,1),(4,4,1),(4,4,1),(4,4,1),(4,2,1)],.7,.25,0
- tie a bg,oc,(1,.5,.5,.5,.5,.5,.5)
- eln; elh

- Command sequence 2: combinations of rests and silences
  - emo mp
  - tmo lg
  - tin a 61
  - *kagan line, set to loop*

    tie r l,[(4,2,0),(4,2,1),(4,2,1),(4,2,0),(4,2,1),(4,2,1),(4,2,0), (4,2,1),(4,2,1),(4,2,0),(4,2,1),(4,2,1)]
  - *accent the first of each articulation*

    tie a bg,oc,(.5,1,.5, .5,.5,.5, .5,.5,.5, .5,.5,.5)
  - *turning on silence mode will use parameters even for rests*

    timode s on
  - tin b 68
  - *create genetic variations using a high crossover, no mutation*

    tie r gr,[(4,2,0),(4,2,1),(4,2,1),(4,2,0),(4,2,1),(4,2,1),(4,2,0),
    (4,2,1),(4,2,1),(4,2,0),(4,2,1),(4,2,1)],1,0,0
  - tie a bg,oc,(.5,1,.5, .5,.5,.5, .5,.5,.5, .5,.5,.5)
  - *turning on silence mode will use parameters even for rests*

timode s on

- eln; elh

- Command sequence 3: multiple rhythmic values:
  - emo mp
  - tmo lg
  - tin a 61
  - *kroboto line, set to loop*

    tie r l,[(4,3,1),(4,1,1),(4,2,1),(4,2,1),(4,1,1),(4,1,1),(4,2,1),
    (4,3,1),(4,1,1),(4,2,1),(4,2,1),(4,1,1),(4,1,1),(4,2,1)]

  - *accent the first of each articulation*

    tie a bg,oc,(1,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5)

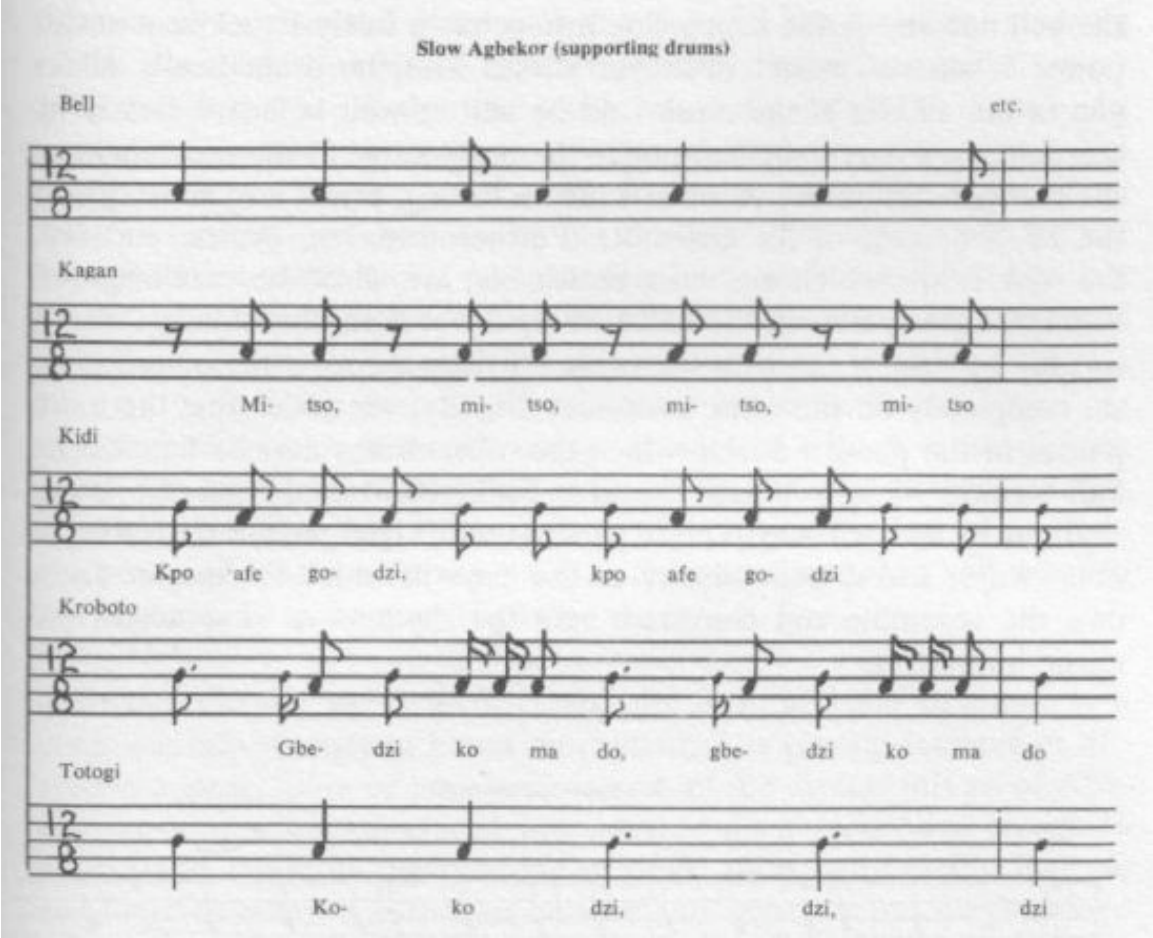
  - tin b 68
  - *create genetic variations using a high crossover and mutation rate and some elitism*

    tie r gr,[(4,3,1),(4,1,1),(4,2,1),(4,2,1),(4,1,1),(4,1,1),(4,2,1),
    (4,3,1),(4,1,1),(4,2,1),(4,2,1),(4,1,1),(4,1,1),(4,2,1)],.9,.25,0.1

  - tie a bg,oc,(1,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5)
  - eln; elh

18.10. Polyphonic African Drum Patterns with a GA

- Slow Agbekor (Chernoff 1979)



- Command sequence:
  - emo mp
  - tmo lg
  - tin a 45
  - tie r gr,[(4,4,1),(4,4,1),(4,2,1),(4,4,1),(4,4,1),(4,4,1),(4,2,1)],.7,.15,0
  - tie a bg,oc,(1,.5,.5,.5,.5,.5,.5)
  - tin b 60
  - *create genetic variations using a high crossover, no mutation*

    tie r gr,[(4,2,0),(4,2,1),(4,2,1),(4,2,0),(4,2,1),(4,2,1),(4,2,0),
    (4,2,1),(4,2,1),(4,2,0),(4,2,1),(4,2,1)],1,0,0
  - tie a bg,oc,(.5,1,.5, .5,.5,.5, .5,.5,.5, .5,.5,.5)

- *turning on silence mode will use parameters even for rests*

  timode s on

- tin c 68

- *create genetic variations using a high crossover and mutation rate and some elitism*

  tie r gr,[(4,3,1),(4,1,1),(4,2,1),(4,2,1),(4,1,1),(4,1,1),(4,2,1),
  (4,3,1),(4,1,1),(4,2,1),(4,2,1),(4,1,1),(4,1,1),(4,2,1)],.9,.25,0.1

- tie a bg,oc,(1,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5)

- eln; elh

18.11. Reading: Biles, GenJam in Perspective: A Tentative Taxonomy for GA Music and Art Systems

- Biles, J. A. 2003. "GenJam in Perspective: A Tentative Taxonomy for GA Music and Art Systems." *Leonardo* 36(1): 43-45.

- What are the alleles and chromosomes in this study?

- At what level of the chromosome do the mutations operate? What types of mutations are used

- How is fitness measured?

- How does the concept of "musically meaningful mutations" deviate from conventional GAs?

- Which does the author suggest are more solution-rich: artistic domains or non-artistic domains?

18.12. GenJam Example

- Video: Demonstration created in 2003

18.13. Reading: Magnus, Evolving electroacoustic music: the application of genetic algorithms to time-domain waveforms

- Magnus, C. 2004. "Evolving electroacoustic music: the application of genetic algorithms to time-domain waveforms." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 173-176.

- What are the alleles and chromosomes in this study?
- What types of mutations were explored in this study?
- Is there a distinction between genotype and phenotype?
- The author writes: "at each stage of programming, choices must be made that introduce designer bias into the system"; it this a problem?

Chapter 19. Meeting 19, Approaches: Grammars and L-Systems

19.1. Announcements

- Sonic system draft due: 27 April
- No class Tuesday, 20 April
- Be sure to do reading for next class:

  Riskin, J. 2003. "The Defecating Duck, or, the Ambiguous Origins of Artificial Life." *Critical Inquiry* 29(4): 599-633.

19.2. Quiz

- 10 Minutes

19.3. String Rewriting Systems

- Given an alphabet and rewrite (production) rules, transform strings
- A wide variety of formalizations and approaches
- Axel Thue: first systematic treatment
- Noam Chomsky: applied concept of re-writing to syntax of natural languages

19.4. Formal Grammars

- A set of rules for a formal language
- Formal grammars can be generative or analytic
- Generative grammars defined by
  - A finite set of nonterminal symbols (variables that can be replaced)
  - A finite set of terminal symbols (constants)
  - An axiom, or initial state
  - A finite set of production rules, replacing variables with variables or constants
- Generative grammars are iterative

19.5. Lindenmayer Systems

- Based on 1968 work of Aristid Lindenmayer
- Origins in model of a natural systems: "a theoretical framework for studying the development of simple multicellular organisms"
- 1984: began use of using computer graphics for visualization of plan structures
- L-systems: formal grammars where re-writing is parallel, not sequential: all symbols are simultaneously replaced

Image: Public domain (Wikipedia)

YouTube (http://www.youtube.com/watch?v=L54SE9KTMSQ)

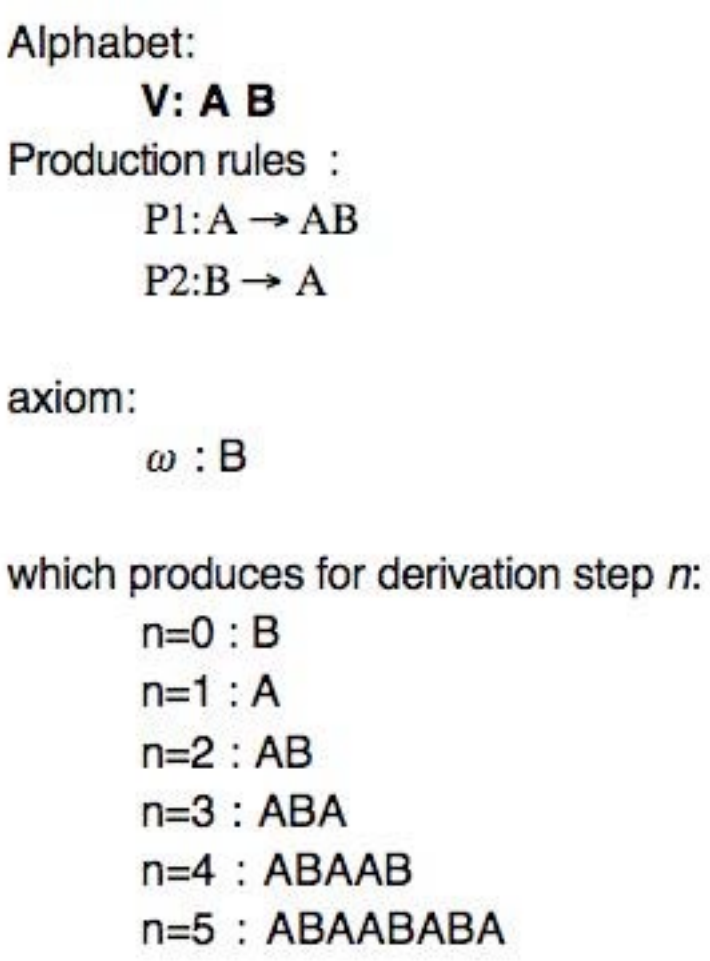
YouTube (http://www.youtube.com/watch?v=t-FZhw9G-RQ)

YouTube (http://www.youtube.com/watch?v=t-FZhw9G-RQ)

- Motivation from natural systems: idea of cell divisions of occurring at the same time
- L-systems can take many different forms depending on rule systems and alphabet components

19.6. Context-Free

- Rules match one source to one or more destination
- Example:



Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

- Originally proposed by Lindenmayer to model growth of algae
- Graphic representation Prusinkiewicz and Lindenmayer (1990)

Figure 1.3: Example of a derivation in a DOL-system.

19.7. Context-Sensitive

- Rules match two or more sources to one or more destination
- 1L systems: match left or right of target source
- 2L systems: match left and right of target source
- 1L systems can be considered 2L systems with an empty (open matching) context
- Example:

Alphabet:

V:ab

Production rules:

P1:b < a > ∅ → b

P2:b → a

axiom:

ω : baaaaaaaa

which produces for derivation step n:

n=0 : baaaaaaaa
n=1 : abaaaaaaa
n=2 : aabaaaaaa
n=3 : aaabaaaaa
n=4 : aaaabaaaa

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

19.8. Non-Deterministic and Table L-systems

- A context-sensitive or context free grammar can be deterministic
- If the application of rules if probabilistic, non-deterministic grammar is created
- Common approach: map one source to two or more destinations, with weighted probabilities for each destination
- Example:

Alphabet :

V: A B

Production rules :

P1: $A \xrightarrow{70\%} AB$

P2: $A \xrightarrow{30\%} BA$

P3: $B \longrightarrow A$

axiom :

ω : A

which can produce for derivation step n:

n=0 : A
n=1 : AB
n=2 : ABA
n=3 : BAAAB
n=4 : ABAABBAA

or:

n=0 : A
n=1 : BA
n=2 : AAB
n=3 : ABABA
n=4 : BAABAAAB

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

- Alternatively, rules can be changed during production, producing a Table L-system (Manousakis 2006, p. 29)

- Example:

Alphabet:

V: A B

axiom:

ω: **B**

Table 1:

Production rules :

$P1: A \rightarrow AB$

$P2: B \rightarrow A$

Table 2:

Production rules :

$P1: A \rightarrow B$

$P2: B \rightarrow BA$

If the set changes on derivation step n=3, this would produce:

T1	n=0 : B
	n=1 : A
	n=2 : AB
T2	n=3 : BBA
	n=4 : BABAB
	n=5 : BABBABBA

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

19.9. Non-Propagative L-systems

- Where rules replace source with more than one successor, the system grows and is propagative
- If rules only encode one-value destinations, the rule system is non-propagative

- Context-sensitive non-propagative L-systems are identical to a standard 1D CA
- Example:

Alphabet :

V : A B

Production rules :

P1: A < A > A → B

P2: A < A > B → A

P3: A < B > A → B

P4: A < B > B → A

P5: B < A > A → A

P6: B < A > B → B

P7: B < B > A → A

P8: B < B > B → B

axiom :

ω: BBBBBBBBBABBBBBBBBB

Using the classic CA visualization for this grammar, and interpreting A = 1 (black pixel) and B = 0 (grey pixel), the first 30 generations look like this:

Figure 2.3. Cellular automata with L-systems.

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

19.10. Musical and Artistic Application of L-systems

- First published implementation: Prusinkiewicz

  Prusinkiewicz, P. 1986. "Score Generation with L-Systems." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 455-457.

- A spatial mapping of 2D graphical output of L-system curves to pitch (vertical) and duration (horizontal)

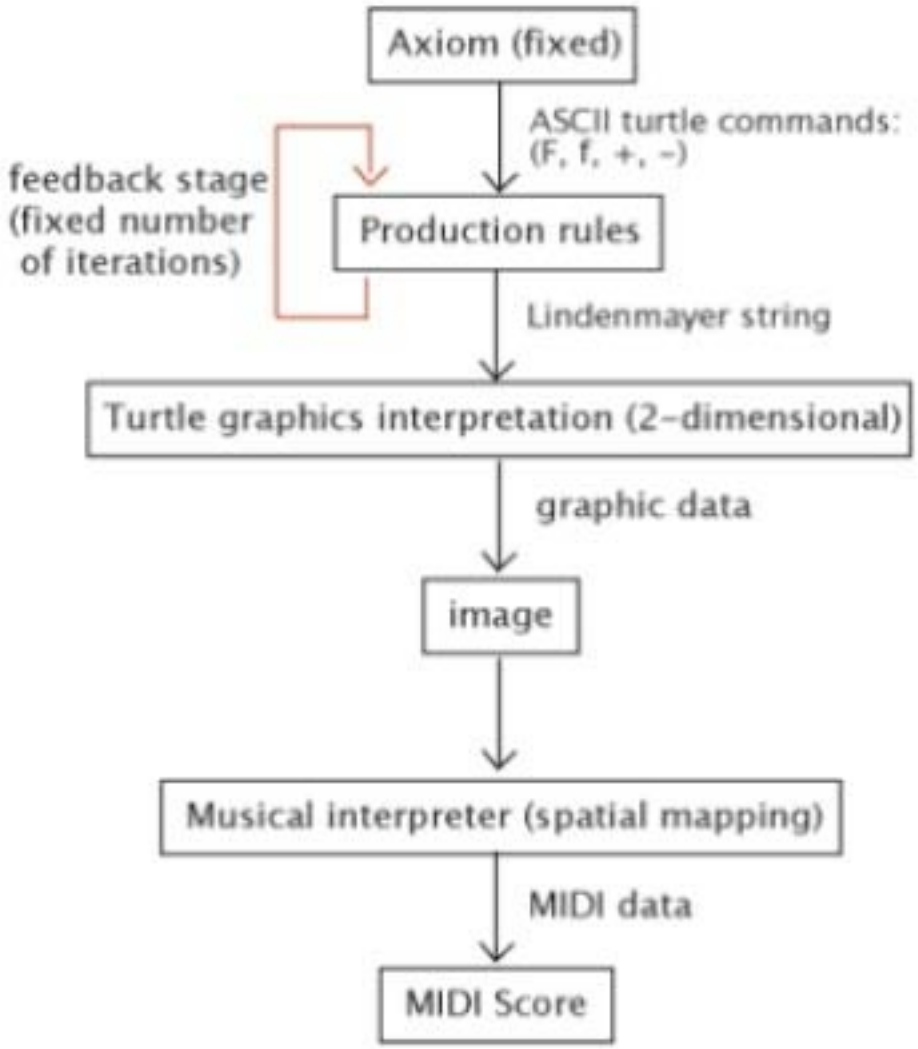
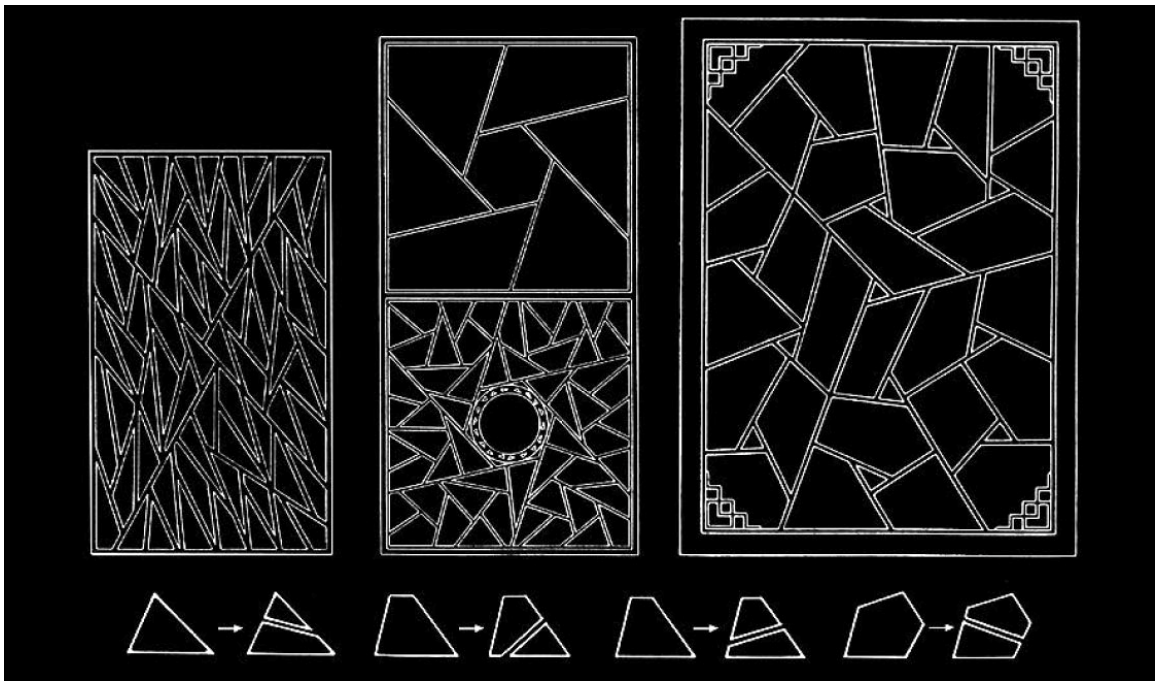


Figure 3.1 Prusinkiewicz' s L-system interpreter concept.

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

- States determine intervals, not absolute values
- Suggest application to other parameters: tempo, amplitude, and position of sound in space
- Creative applications in the visual arts and architecture

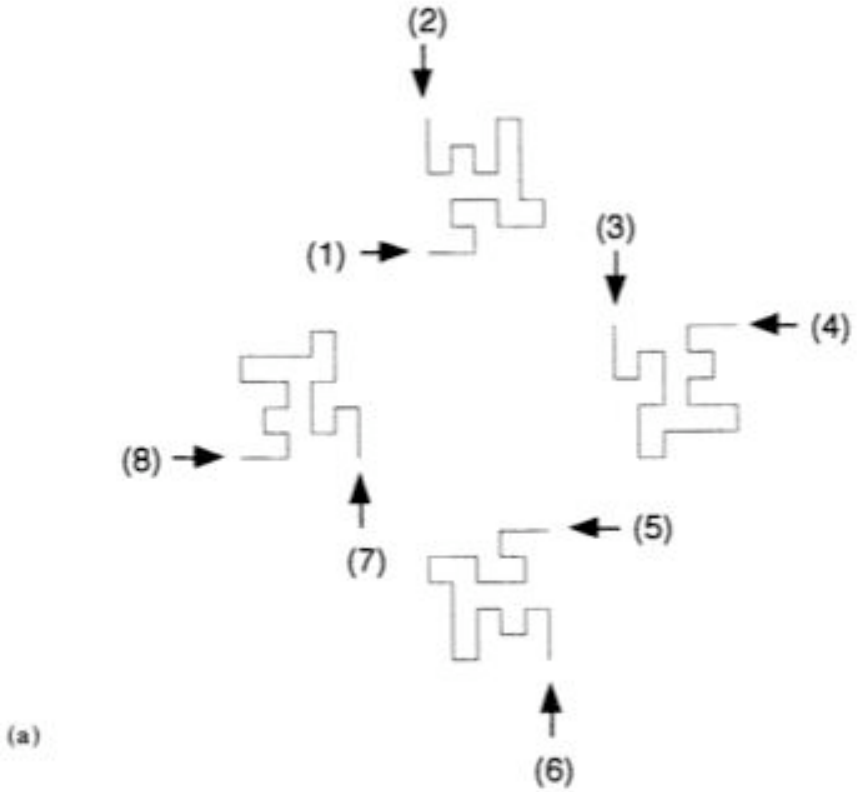
Stiny, G. and J. Gips. 1972. "Shape Grammars and the Generative Specification of Painting and Sculpture." In *Information Processing 71*. C. V. Freiman, ed. Amsterdam: North Holland. 1460-1465. Internet: http://www.shapegrammar.org/ifip/.



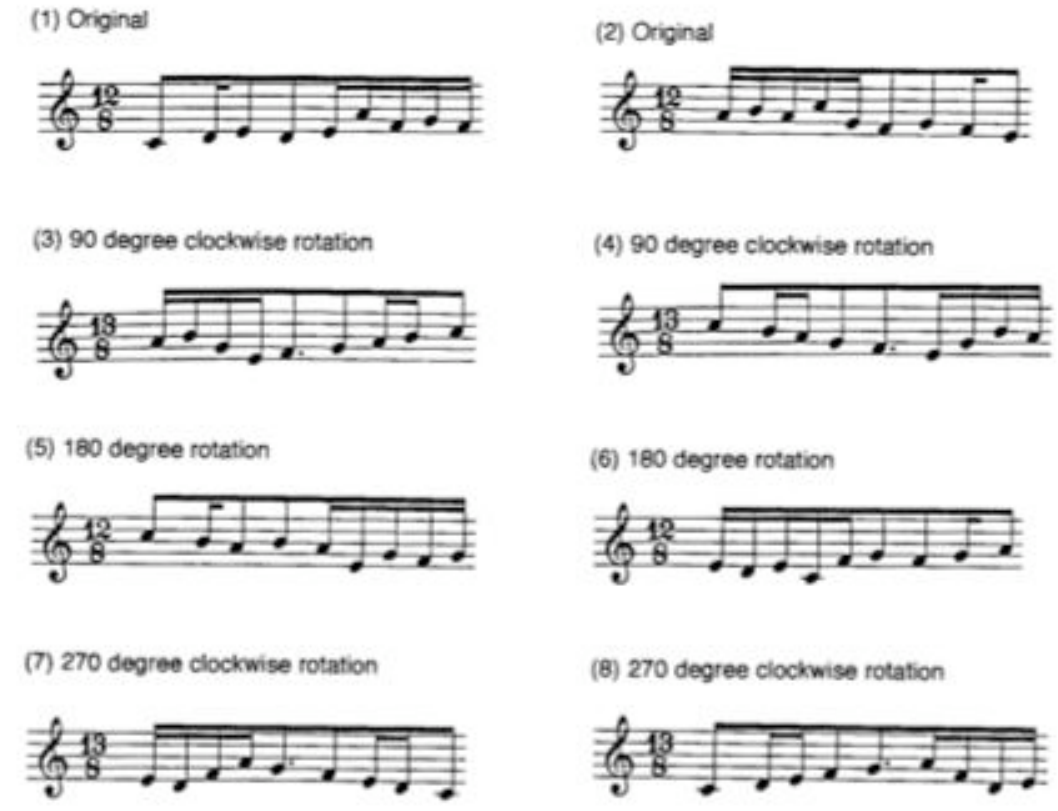
Courtesy of George Stiny. Used with permission.

19.11. Reading: Mason and Saffle

- Mason, S. and M. Saffle. 1994. "L-Systems, Melodies and Musical Structure." *Leonardo Music Journal* 4: 31-38.
- Are deterministic CA always fractal?
- The basic mapping (after Prusinkiewicz)



(a)



(b)

Source: Mason, S. and M. Saffle. "L-Systems, Melodies and Musical Structure." *Leonardo Music Journal* 4 (1994): 31-38.

- What are some alternative ways the 2D turtle graphics can be mapped and as musical values?
- Is it significant that "any melody can be modeled with an L-system, including the songs of aboriginal hunters, the plainchants of the medieval christian liturgy, the themes of beethoven's symphonies and popular song tunes," as the authors claim?
- What is the implied connection between fractals and beauty. Is this connection sufficiently supported?

19.12. A Grammar Specification String and Python Implementation

- Define a grammar in two required parts: alphabet and rules

  Both are specified in key{value} pairs

  Rules are specified as source{destination} pairs

  ```
  a{3}b{-2} @ a{b} b{a}
  ```

- Optionally include the axiom (one chosen at random otherwise)

  ```
  a{3}b{-2} @ a{b} b{a} @ baba
  ```

- Permit one to many rules of any size

  ```
  a{3}b{-2} @ a{ba} b{abb} @ baba
  ```

- Permit context sensitivity as many to one or many to many rules (not yet implemented)

  ```
  a{3}b{-2} @ aa{ba} bab{abb} @ baba
  ```

- Match any source as pattern specified with quasi regular expressions (not yet implemented)

  ```
  a{3}b{-2} @ *aa{ba} b*b{abb} bb*{abb} @ baba
  ```

- Configure non-deterministic destinations as two or more weighted options

  Weights can be specified with a floating point or integer value following the destination

  ```
  a{3}b{-2} @ a{ba|ab} b{abb=3|aa=2} @ baba
  ```

- Can create a grammar instance and view step-wise output

```
>>> from athenaCL.libATH import grammar
>>> g = grammar.Grammar()

>>> g.load('a{3}b{-2} @ a{b} b{a} @ baba')
>>> g.next(); g.getState()
'baba'
>>> g.next(); g.getState()
'abab'
>>> g.next(); g.getState()
'baba'
```

```
>>> g.load('a{3}b{-2} @ a{ba|ab} b{abb=3|aa=2} @ baba')
>>> g.next(); g.getState()
'abbababbba'
>>> g.next(); g.getState()
'baaaabbababbbaabbabbaaba'
>>> g.next(); g.getState()
'aaabbaabbaabbaabaabbabaaaaabbbaabababbabbbaabbabbbabaabbba'
```

- Can translate grammar string back into a list of source values

```
>>> from athenaCL.libATH import grammar
>>> g = grammar.Grammar()
>>> g.load('a{3}b{-2} @ a{ba|ab} b{abb=3|aa=2} @ baba')
>>> g.next(); g.getState()
'abbababbba'
>>> g.getState(values=True)
[3.0, 3.0, 3.0, -2.0, 3.0, -2.0, -2.0, -2.0, 3.0]
```

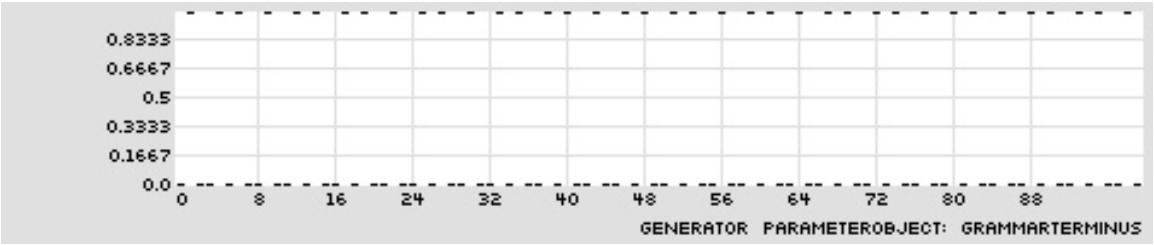
19.13. Grammar as ParameterObject

- The grammarTerminus ParameterObject

```
:: tpv grammar
Generator ParameterObject
{name,documentation}
grammarTerminus     grammarTerminus, grammarString, stepCount, selectionString
                    Description: Produces values from a one-dimensional string
                    rewrite rule, or generative grammar. The terminus, or final
                    result of the number of generations of values specifed by
                    the stepCount parameter, is used to produce a list of
                    defined values. Values are chosen from this list using the
                    selector specified by the selectionString argument.
                    Arguments: (1) name, (2) grammarString, (3) stepCount, (4)
                    selectionString {“randomChoice”, “randomWalk”,
                    “randomPermutate”, “orderedCyclic”,
                    “orderedCyclicRetrograde”, “orderedOscillate”}
```

- The Lindenmayer algae model after 10 generations

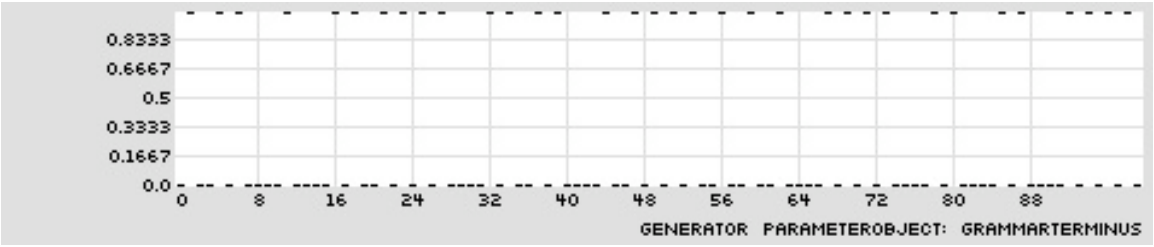
```
:: tpmap 100 gt,a{0}b{1}@a{ab}b{a}@b,10,oc
grammarTerminus, a{0}b{1}@a{ab}b{a}@b, 10, orderedCyclic
TPmap display complete.
```



- Modified Lindenmayer algae model after 10 generations with non-deterministic rule variation

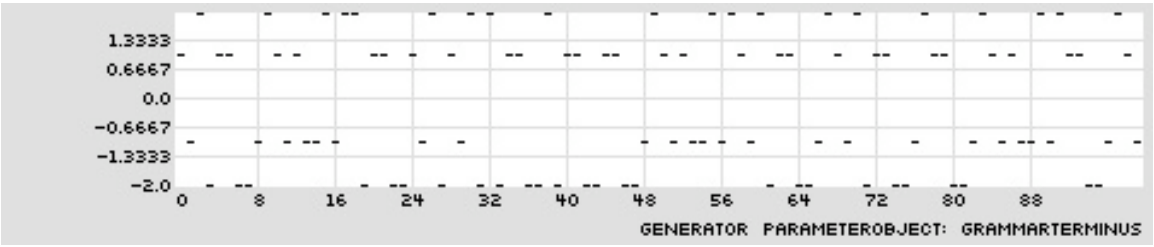
```
:: tpmap 100 gt,a{0}b{1}@a{ab}b{a|aaa}@b,10,oc
```

```
grammarTerminus, a{0}b{1}@a{ab}b{a=1|aaa=1}@b, 10, orderedCyclic
TPmap display complete.
```



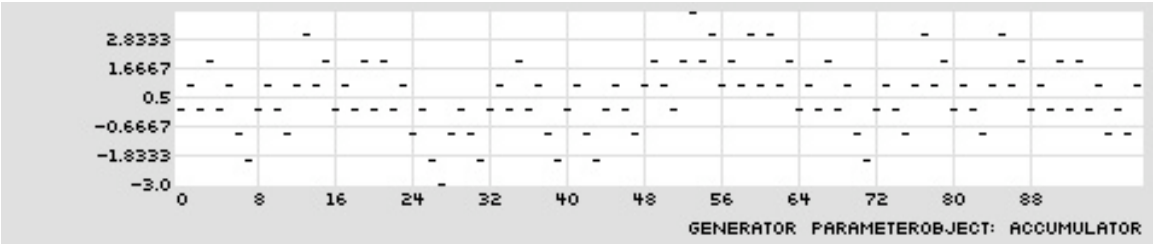
- Four state deterministic grammar

```
:: tpmap 100 gt,a{1}b{-1}c{2}d{-2}@a{ab}b{cd}c{aadd}d{bc}@ac,10,oc
grammarTerminus, a{1}b{-1}c{2}d{-2}@a{ab}c{aadd}b{cd}d{bc}@ac, 10,
orderedCyclic
TPmap display complete.
```



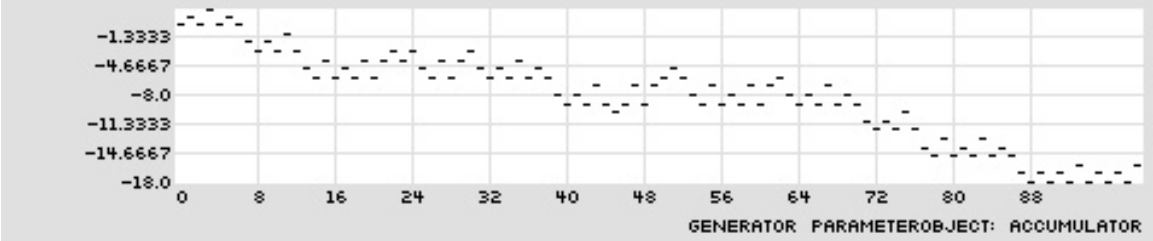
- Four state deterministic grammar placed in an Accumulator PO

```
:: tpmap 100 a,0,(gt,a{1}b{-1}c{2}d{-2}@a{ab}b{cd}c{ad}d{bc}@ac,10,oc)
accumulator, 0, (grammarTerminus, a{1}b{-1}c{2}d{-2}@a{ab}c{ad}b{cd}d{bc}@ac,
10, orderedCyclic)
TPmap display complete.
```



- Four state non-deterministic grammar placed in an Accumulator PO

```
:: tpmap 100 a,0,(gt,a{1}b{-1}c{2}d{-2}@a{ab}b{cd}c{ab|ca}d{ba|db}@ac,10,oc)
accumulator, 0, (grammarTerminus,
a{1}b{-1}c{2}d{-2}@a{ab}c{ab=1|ca=1}b{cd}d{ba=1|db=1}@ac, 10, orderedCyclic)
TPmap display complete.
```



- Alternative approaches to PO interface?
- Mappings and applications in athenaCL?

19.14. Grammar States as Accent Patterns

- Can treat the grammar alphabet as parameter values: integers, floating point values
- Command sequence:
  - emo mp
  - tmo lg
  - tin a 60
  - *non deterministic binary algae generator applied to accent*

    tie r pt,(c,8),(c,1),(gt,a{0}b{1}@a{ab}b{a|aaa}@b,10,oc)
  - tie a c,1
  - *four state deterministic applied to pulse multiplier*

    tie r pt,(c,8), (gt,a{1}b{2}c{4}d{8}@a{ab}b{cd}c{aadd}d{bc}@ac,10,oc),(c,1)
  - *four state deterministic applied to amplitude with different start string*

    tie a gt,a{.25}b{.5}c{.75}d{1}@a{ab}b{cd}c{aadd}d{bc}@bbc,6,oc
  - *four state deterministic applied to transposition with different start string*

    tie f gt,a{0}b{1}c{2}d{3}@a{ab}b{cd}c{aadd}d{bc}@dc,6,oc
  - *four state non-deterministic applied to transposition with different start string*

    tie f gt,a{0}b{1}c{2}d{3}@a{ab}b{cd|aa}c{aadd|cb}d{bc|a}@dc,6,oc
  - eln; elh

19.15. Grammar States as Pitch Values

- Can treat the grammar alphabet as specific pitch values
- Command sequence:
  - emo m
  - tmo lg
  - tin a 32
  - *four state deterministic applied to pulse multiplier*

    tie r pt,(c,8), (gt,a{1}b{2}c{4}d{8}@a{ab}b{cd}c{aadd}d{bc}@ac,8,oc),(c,1)
  - tie o c,-2
  - *four state deterministic applied to transposition with different start string*

    tie f gt,a{0}b{7}c{8}d{2}@a{ab}b{cd}c{aadd}d{bc}@ad,6,oc
  - *four state deterministic applied to amplitude with different start string*

    tie a gt,a{.25}b{.5}c{.75}d{1}@a{ab}b{cd}c{aadd}d{bc}@bbc,6,oc
  - eln; elh

19.16. Grammar States as Pitch Transpositions

- Can treat the grammar alphabet as transpositions iteratively processed through an Accumulator
- Command sequence:
  - emo m
  - tmo lg
  - tin a 15
  - *four state deterministic applied to pulse multiplier*

    tie r pt,(c,8), (gt,a{1}b{2}c{4}d{8}@a{ab}b{cd}c{aadd}d{bc}@ac,8,oc),(c,1)
  - *four state deterministic applied to accumulated transposition with different start string*

    tie f a,0,(gt,a{1}b{-1}c{7}d{-7}@a{ab}b{cd}c{ad}d{bc}@ac,10,oc)
  - *four state deterministic applied to amplitude with different start string*

tie a gt,a{.25}b{.5}c{.75}d{1}@a{ab}b{cd}c{aadd}d{bc}@bbc,6,oc

- eln; elh

19.17. Grammar States as Path Index Values

- Can treat the grammar alphabet as index values from the Path iteratively processed through an Accumulator
- Command sequence:
  - emo m
  - *create a single, large Multiset using a sieve*

    pin a 5@0|7@2,c2,c7
  - tmo ha
  - tin a 6
  - *constant rhythm*

    tie r pt,(c,4),(c,1),(c,1)
  - *select only Multiset 0*

    tie d0 c,0
  - *select pitches from Multiset using accumulated deterministic grammar starting at 12*

    tie d1 a,12,(gt,a{1}b{-1}c{2}d{-2}@a{ab}b{cd}c{ad}d{bc}@ac,10,oc)
  - *create only 1 simultaneity from each multiset; create only 1-element simultaneities*

    tie d2 c,1; tie d3 c,1
  - *four state deterministic applied to amplitude with different start string*

    tie a gt,a{.25}b{.5}c{.75}d{1}@a{ab}b{cd}c{aadd}d{bc}@bbc,6,oc
  - eln; elh

Chapter 20. Meeting 20, History: Mechanical Musical Automata

20.1. Announcements

- Sonic system draft due: 27 April

20.2. Quiz Review

- ?

20.3. Android

- First usage dates from 1727-51

Image from etymology dictionary removed due to copyright restrictions.

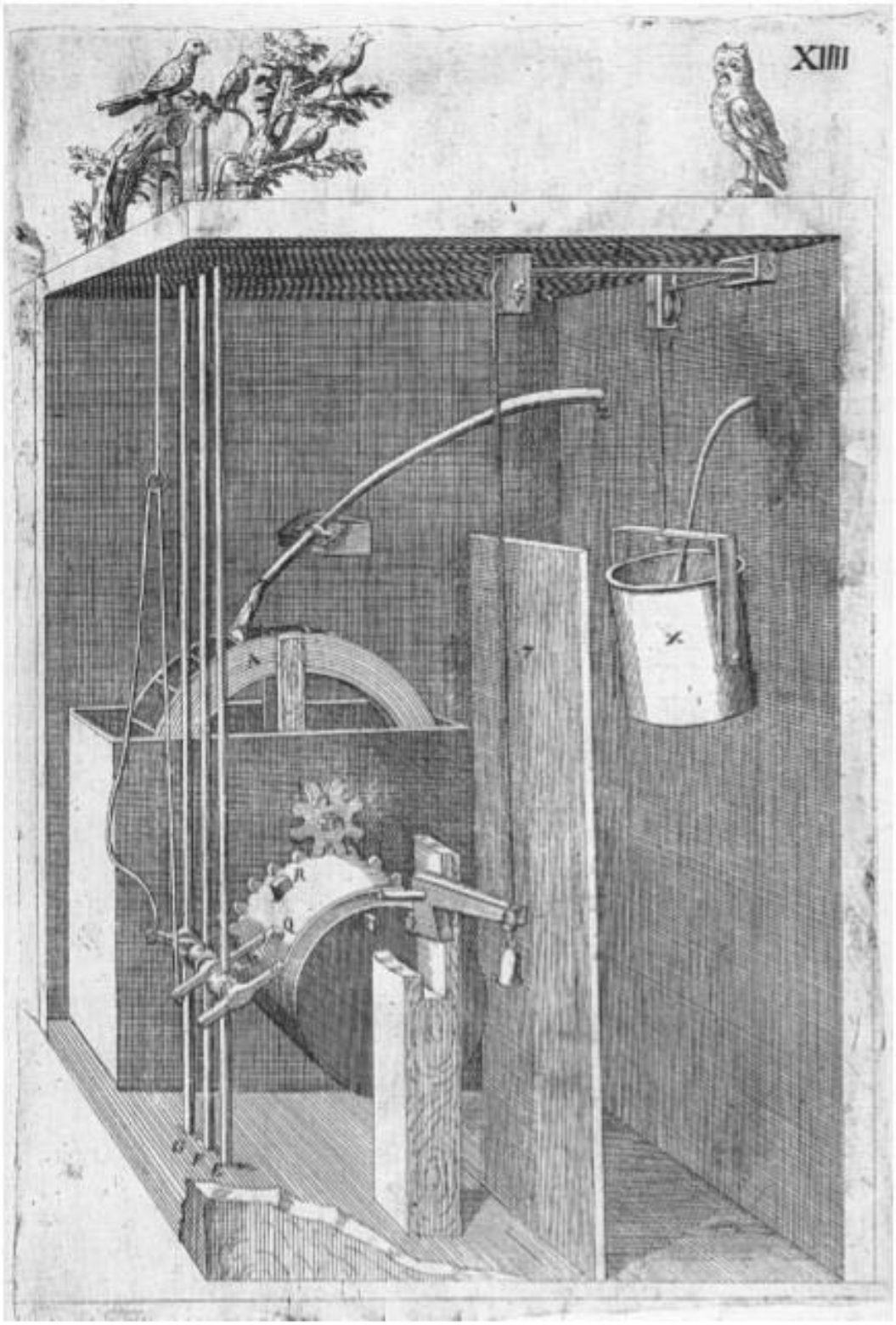
20.4. Mechanical Automata

- Jacques Vaucanson (1709-1782)
- Pierre Jaquet-Droz (1721-1790)
- Wolfgang von Kempelen (1734-1804)
- Joseph Marie Jacquard (1752-1834)
- Charles Babbage (1791-1871)

20.5. Automata: Background

- 18th century mechanical theaters built within clock works
- Often religious scenes of Madonna and Child
- Inspired a wide range of automatic entertainment machines
- Resulted in complex machines and technological advances
- Example: automaton monk from 1560

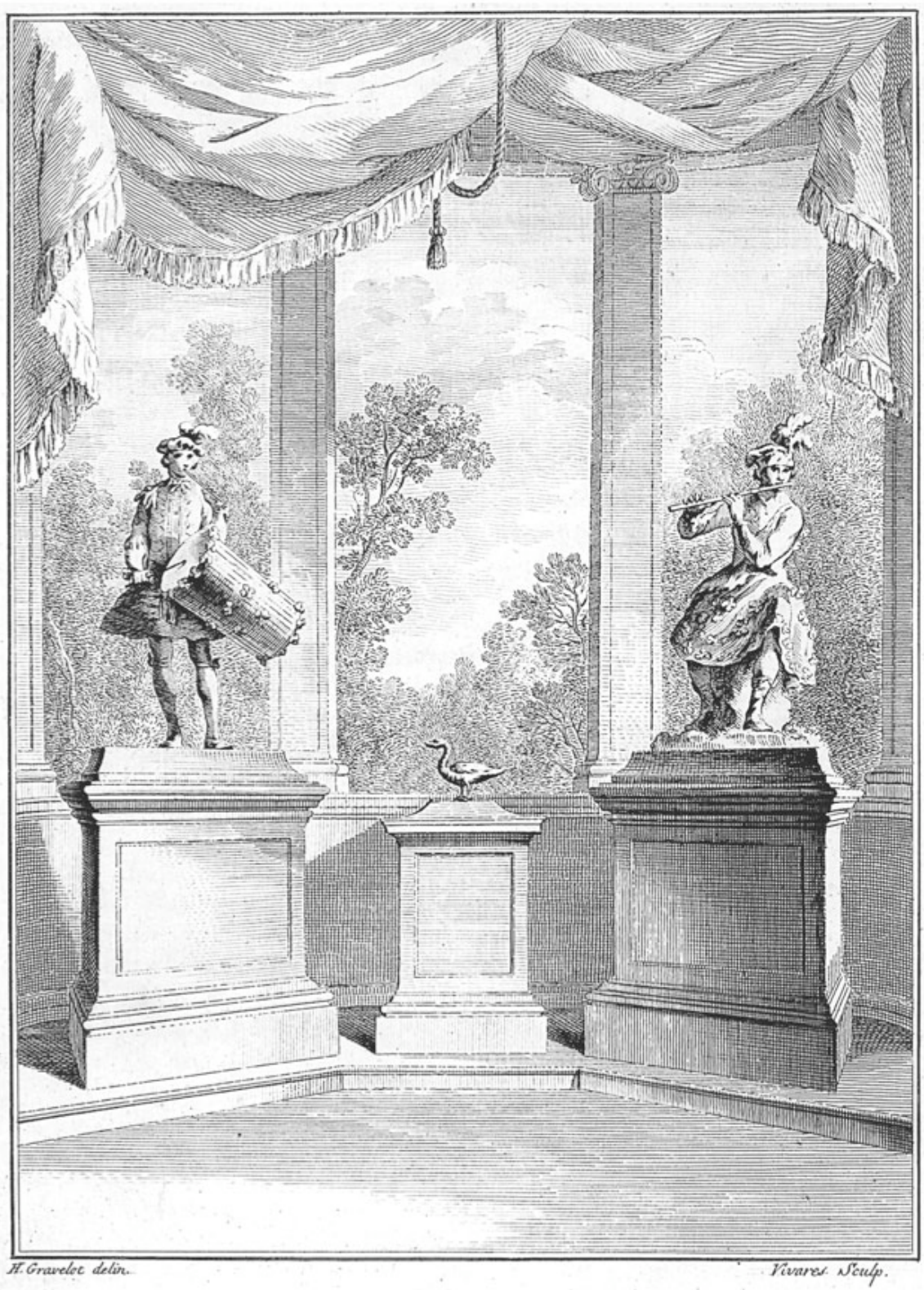
  YouTube (http://www.youtube.com/watch?v=Ycyj76VPOtc)
- Example: automaton of French engineer Isaac de Caus



Isaac de Caus's threatening owl and intimidated birds. (Public domain image)

20.6. Vaucanson: The Flute Player

- Was a Jesuit but renounced religious life for pursuit of automata (Standage 2002)
- Displayed in 1737 (Standage 2003)
- Based on a statue by Antoine Coysevox (2003, p. 613)
- A theory of sound production of the flute
- Acoustically produced sound: different flutes could be substituted
- Concerned with three parameters: air pressure, shape of aperture, length of flute (2003, p. 615)



Public domain image.

Image removed due to copyright restrictions.
See diagram "Fluter de Vaucanson" in Doyon, A., and L. Liaigre.
Jacques Vaucanson, mécanicien de génie. Presses Universitaires de France, 1966.

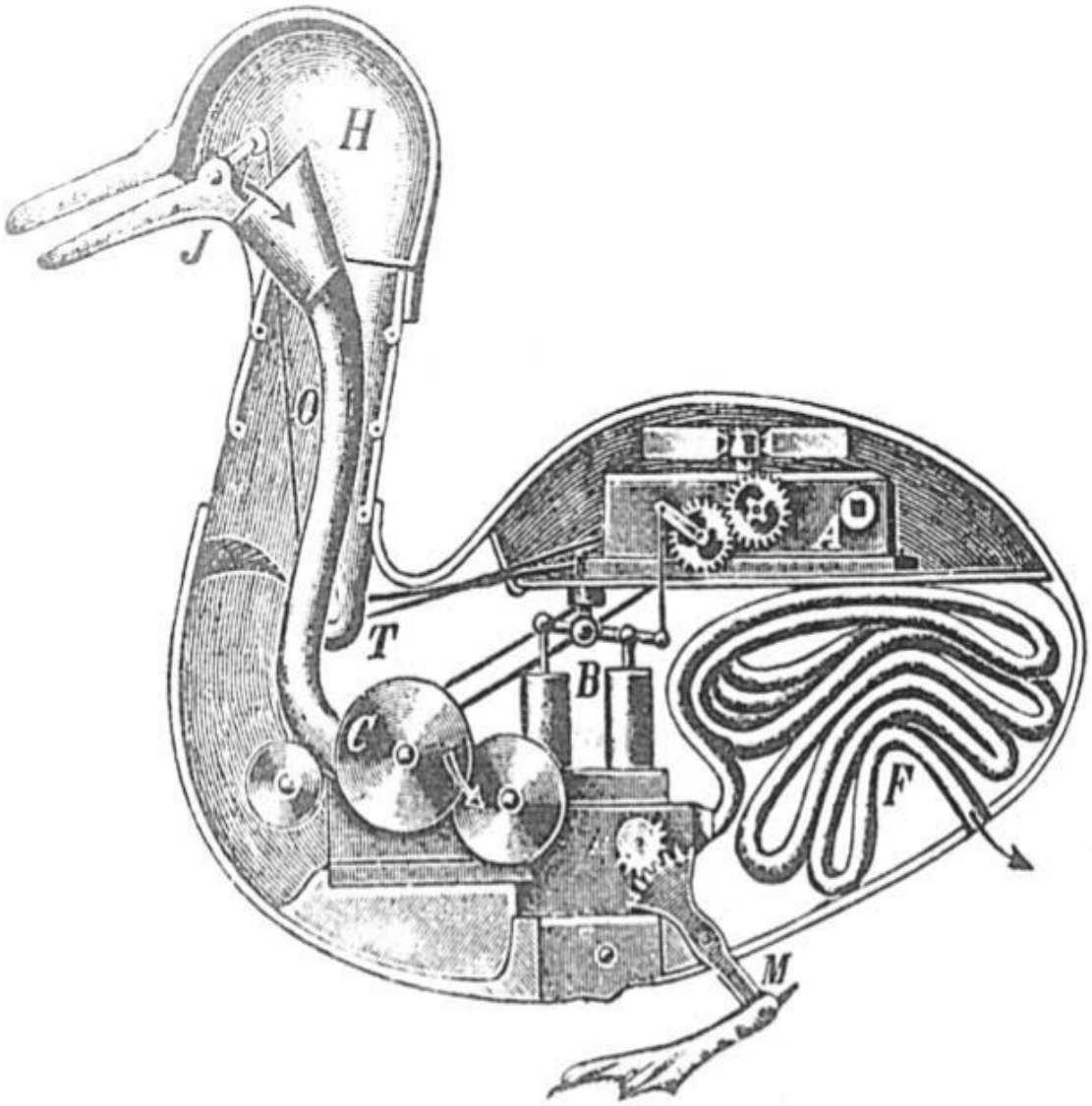
- Used studs on a cylinder to encode data

20.7. Vaucanson: Pipe and Tabor Player

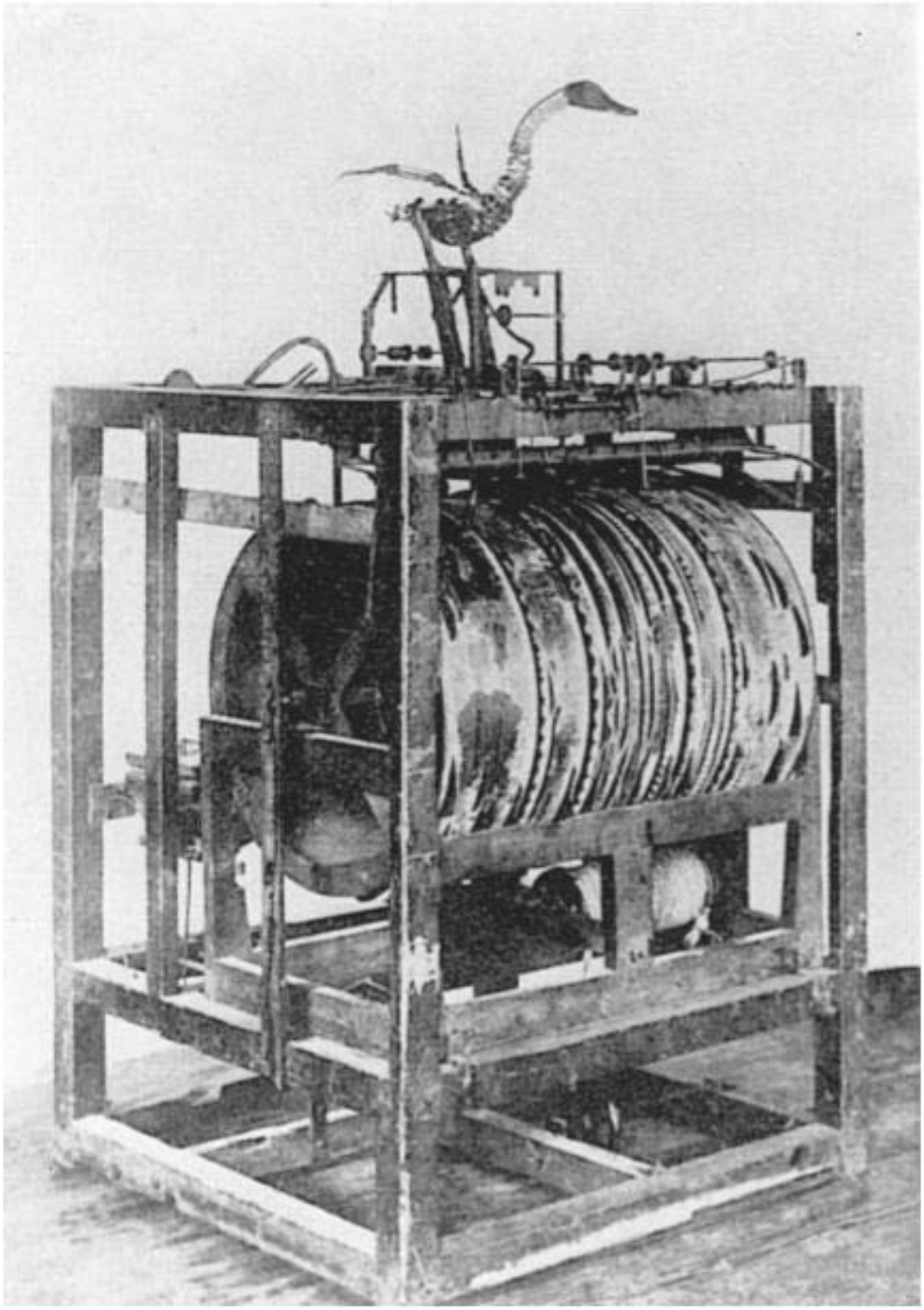
- Displayed in 1738
- Vaucanson "discovered that the blowing pressure for a given note depended upon the preceding note, so that it required more pressure to produce D after an E than after a C..." (2003, p. 616)

20.8. Vaucanson: Defecating Duck

- Displayed in 1738



A mechanical digesting duck, as imagined by a 19th century illustrator. (Public domain image)



Mysterious photo discovered at the Musee des Arts de Metiers in Paris, labeled "Views of Vaucason's Duck received from Dresden." (Public domain image)

YouTube: 3:28 (http://www.youtube.com/watch?v=Pd_21_pfSRo)

- Powered by weight that, with gravity, drove a cylinder
- Part of later projects in moving anatomies: mechanical models of bodily processes such as respiration and circulation: intended for physiological experimentation and to test medical therapies (2003, p. 625)

20.9. Vaucanson: Automated Loom

- Created in 1747

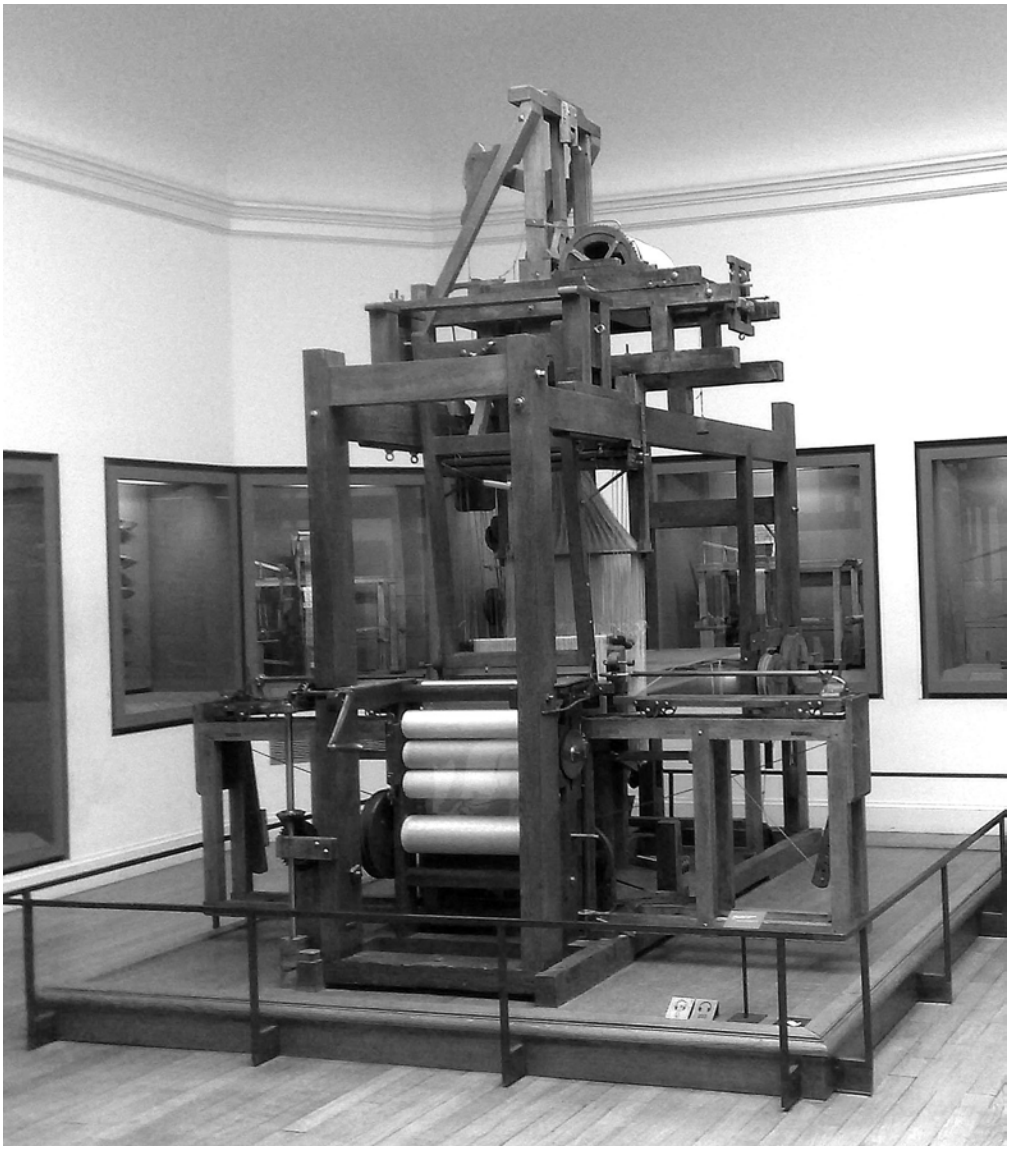
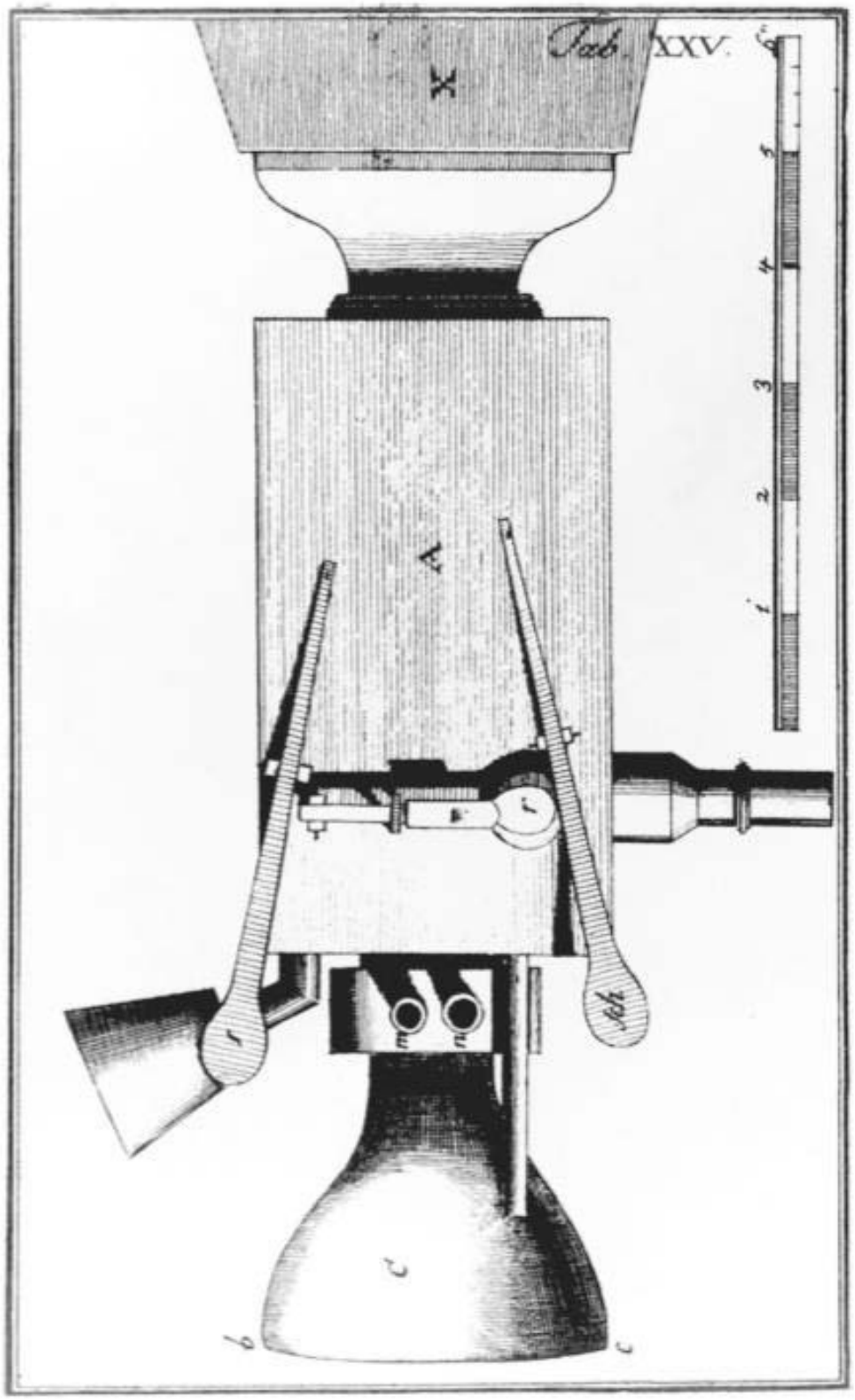


Photo courtesy of bjepson on Flickr.

- Resulted from an analysis of what could be automated in the production of silk
- Did not recreate the same process used by humans
- Task formerly known to take intelligence (reading patterns) was demoted; process that was not known to take intelligence (silk reeling) was promoted

20.10. Kempelen: Speaking Machine

- Kempelen was familiar with Vaucanson's work (Standage 2003)
- Published description of speaking machine in 1791



Kempelen's speaking machine. (Public domain image)

(Public domain image)

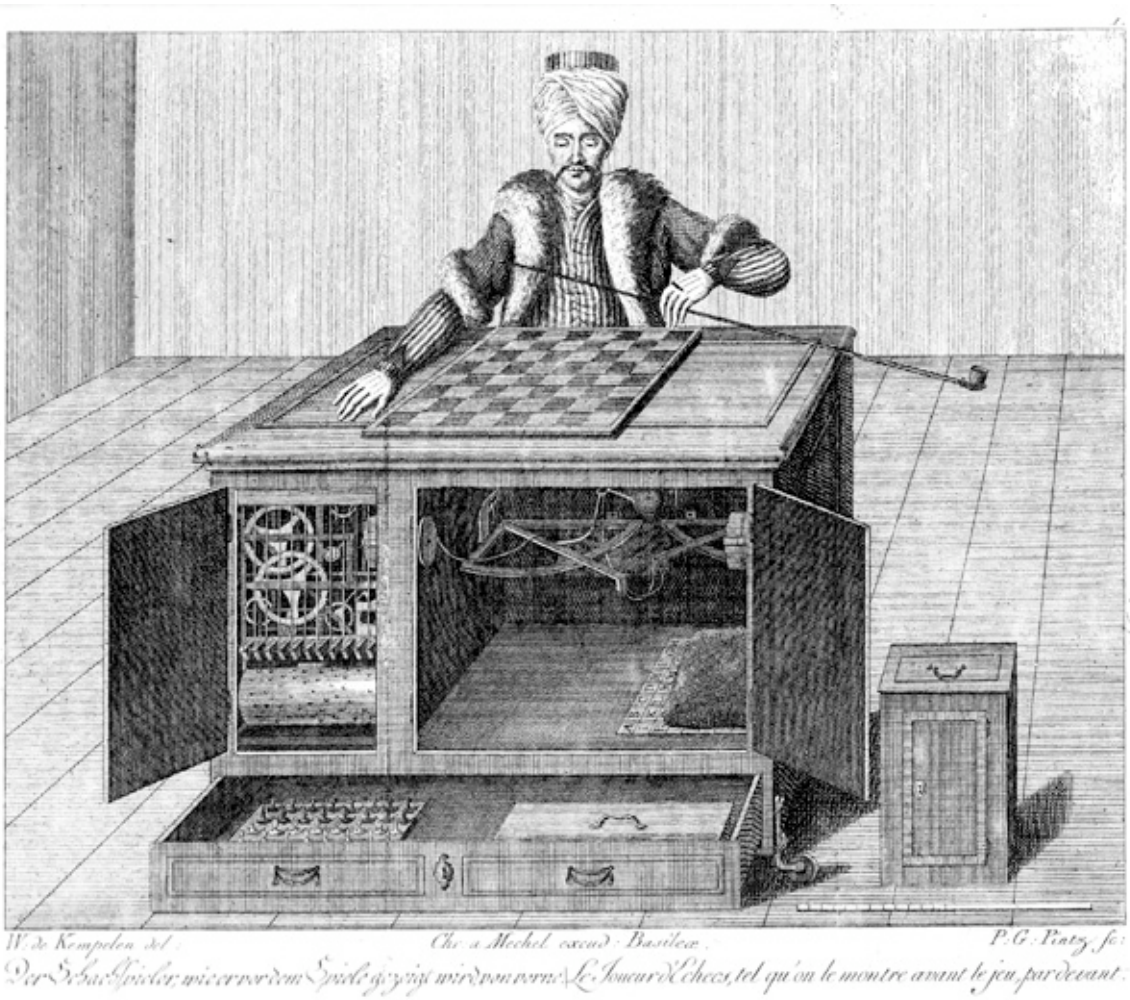
- Part of a tradition of "speaking heads"; Christian Kratzenstein built models of the vocal tract in 1779
- Used bellows, reeds, and models of lungs and mouth: known to say "mama" and "papa"

  YouTube (http://www.youtube.com/watch?v=zYRVqrfY3tQ)
- Kempelen "reported that he had first tried to produce each sound in a given word or phrase independently but failed because the successive sounds needed to take their shape from one another" (2003, p. 619)

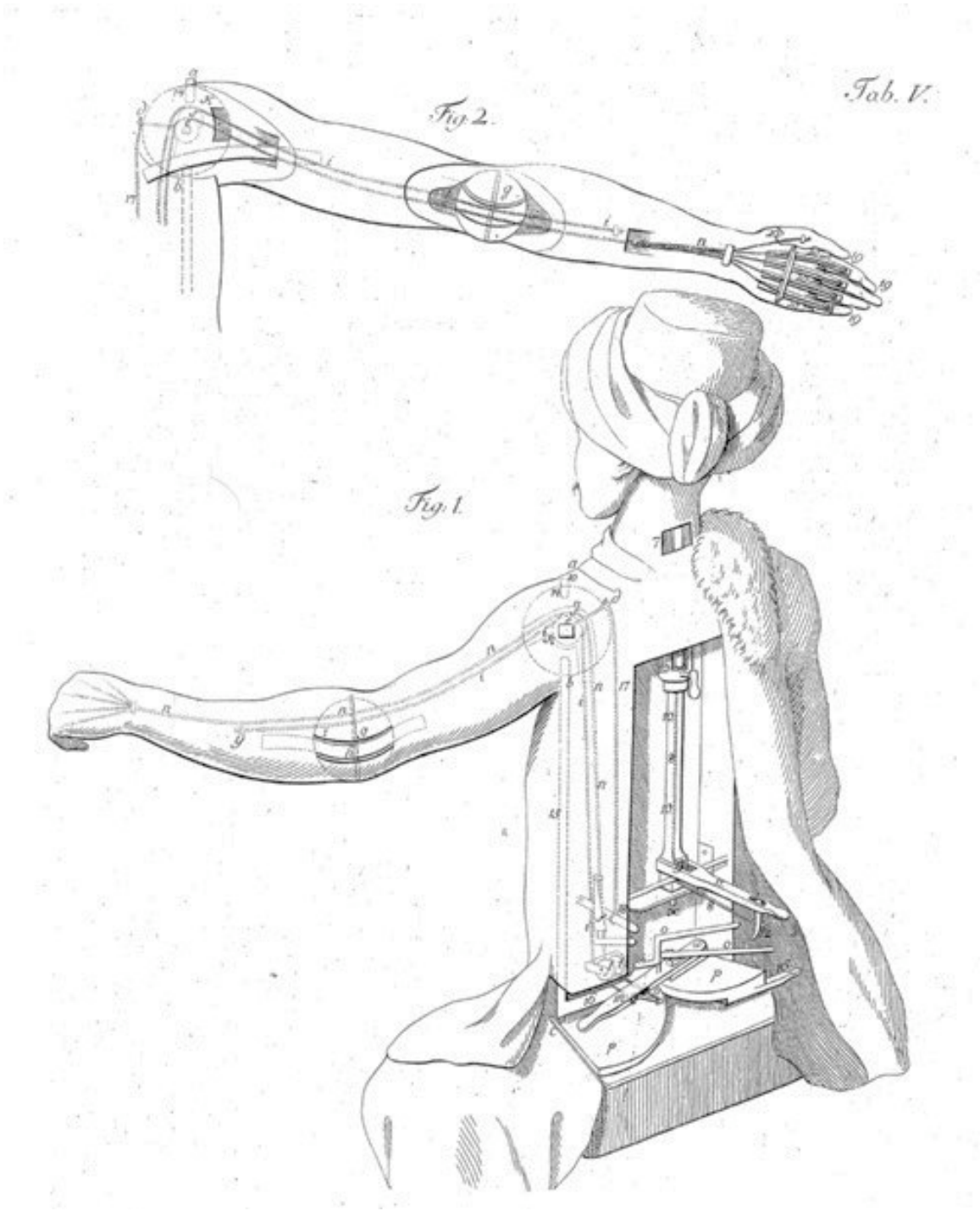
20.11. Kempelen: The Turk

- Kempelen, after seeing a magician at the court of the empress of Austria-Hungary, declared that he could do better; was given six months leave (2003)

- Spring of 1770 returned with the Turk



(Public domain image)



(Public domain image)

- Had an elaborate presentation of the inside of the machine, used a candle to show panels, turned machine around
- Placed two candelabras on top of the cabinet
- Turned a key and made loud, mechanical noises

- Beat most opponents in less than half an hour
- First tour of Europe; in Paris it played Benjamin Franklin and lost to Philidor
- Owned and exhibited by Johann Mälzel from 1808 to 1828 in Milan, Paris, Uniter Kingdom, New York City, Boston, Philadelphia, and Baltimore
- Would play chess, perform end-games (the Knight's Tour), would answer audience questions by point to a board
- Destroyed in a fire in a Philadelphia museum in 1854
- Documentaries and other programs

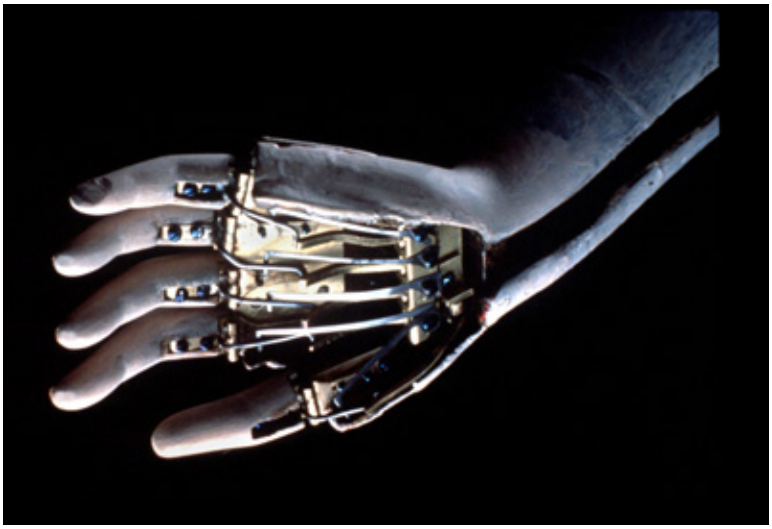
  YouTube (http://www.youtube.com/watch?v=RdT4yG8wczQ)

  YouTube (http://www.youtube.com/watch?v=K3U83LnwMCc)

20.12. Jaquet Droz: The Harpsichord Player

- Built between 1768 and 1774
- Henri Louis Jaquet-Droz builds a harpsichord player machine

- Programmed with studs on a cylinder
- Family used designs to construct prosthetic limbs (Riskin 2003, p. 625)



- Part of a collection of three automata: the writer, the drawer, and the musician

  YouTube (http://www.youtube.com/watch?v=Pd_21_pfSRo)

- Other Jaquet-Droz Devices:

  Singing Bird Box:

  YouTube (http://www.youtube.com/watch?v=HjLy0zausRU)

20.13. Jacquard: Automatic Loom

- Studied Vaucanson's loom in Paris
- Built in 1801

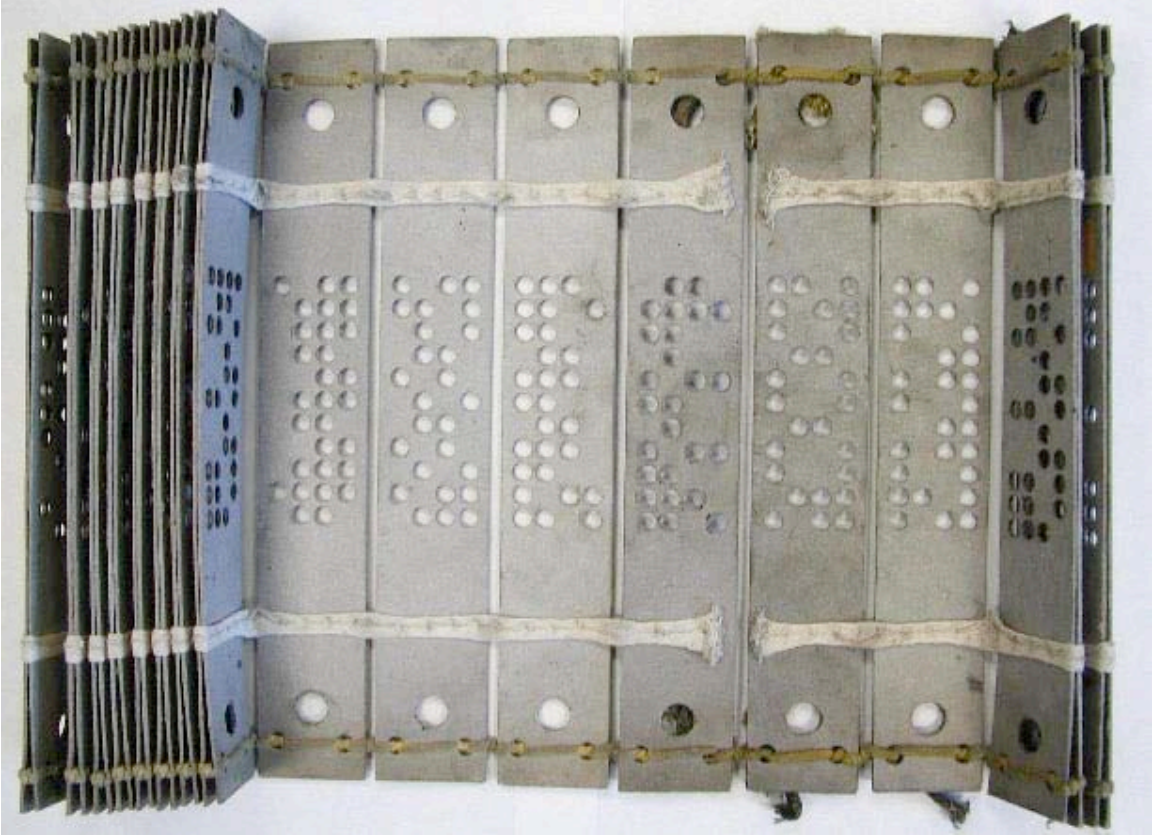


Photo courtesy of Douglas W. Jones at the University of Iowa.

(Public domain image)

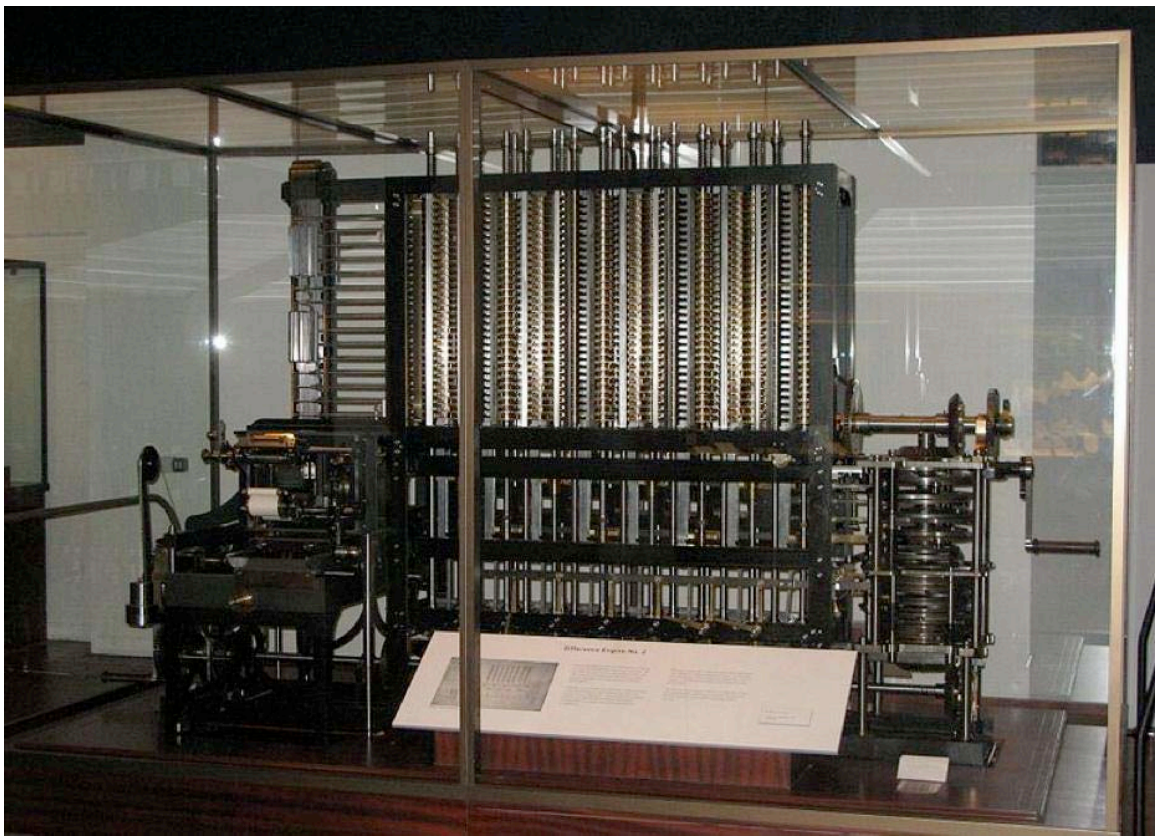
- Loom operations could be stored and recalled later
- Multiple cards could be strung together
- Based on technologies of numerous inventors from the 1700s, including Jacques Vaucanson

20.14. Babbage: The Difference Engine

- Babbage had played chess against the Turk, but suspected it was under human control (Standage 2003)
- Sketched out plan for mathematical automaton in 1821, shortly after playing the Turk (Standage 2003)
- Produced improved design between 1847 and 1849
- Essentially columns that store integers; columns can add value of column n+1 to column n to produce a new value for n
- Automatic calculation of polynomial functions with Newton's method of divided differences: finding value that must be added to obtain the next result:

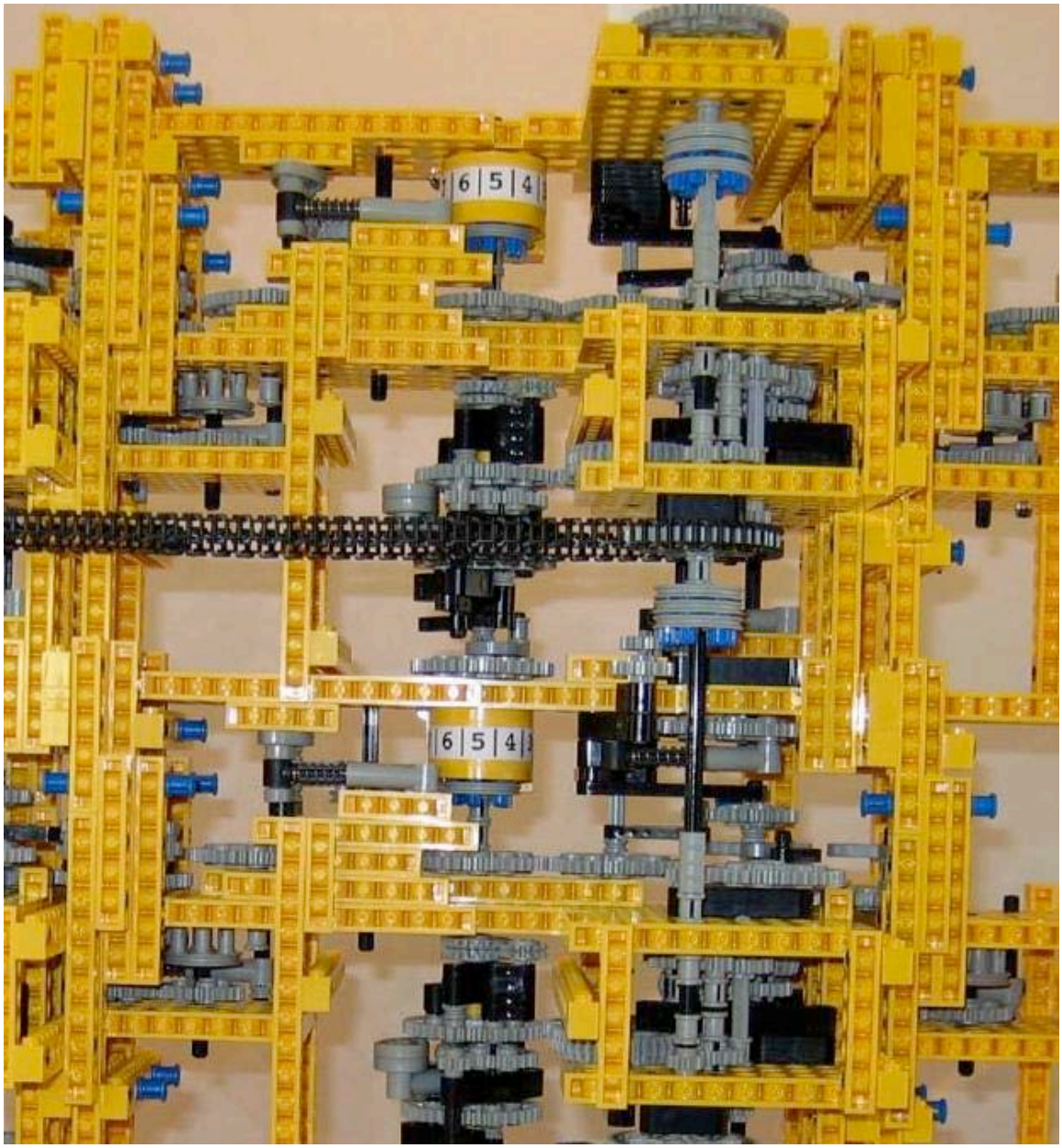
x	f(x) $2x^2 + 3x + 5$	First Difference	Second Difference
1	10	9	4
2	19	13	4
3	32	17	4
4	49	21	
5	70		

- 25,000 parts, 15 tons, 8 feet high: never completed
- Employed rotating cylinders that stored a single digit
- 1989-1991: Difference Engine No 2 constructed and used



- A difference engine built of Lego by Andrew Carol

http://acarol.woz.org



Courtesy of Andrew Carol. Used with permission.

20.15. Babbage: The Analytical Engine

- Continued working on until death in 1871
- Machine that could be programmed with punched cards
- Loops of punched cards such as those used in the Jacquard loom
- Ada Lovelace designed as program for the Analytical Engine

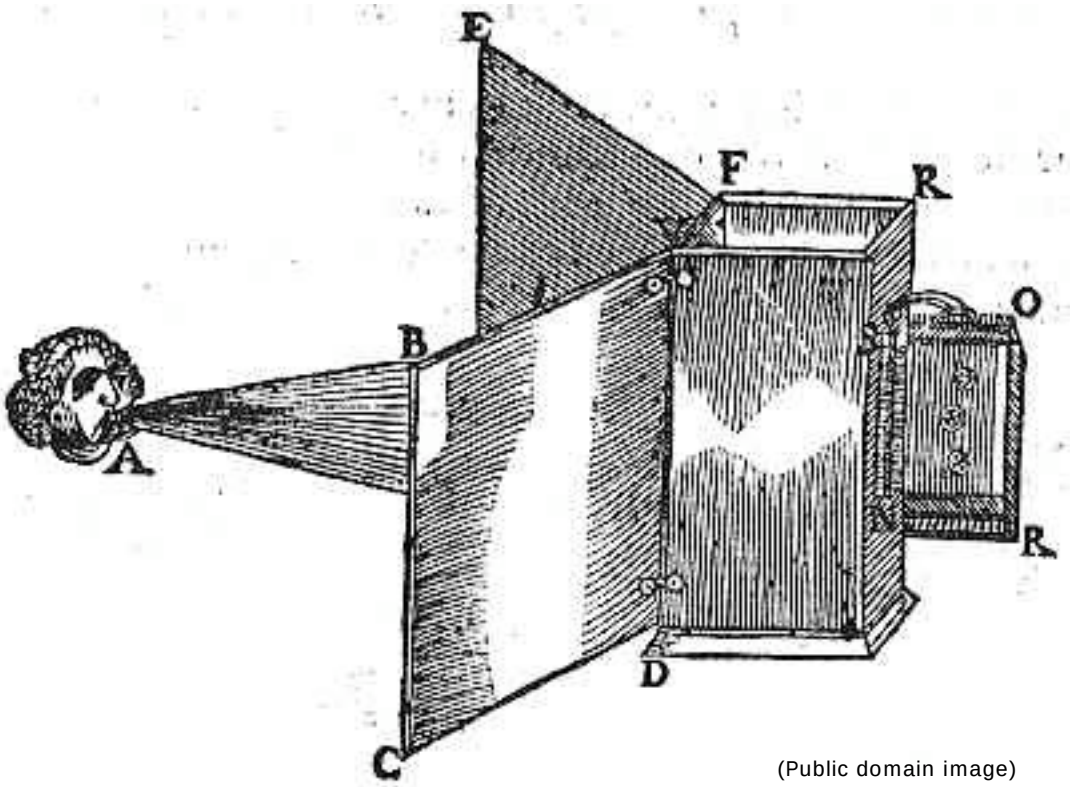
- Daughter of Lord Byron
- Translated article on the Analytical Engine by Luigi Menabrea added notes that included algorithm
- Suggested in 1843: if "the fundamental relations of pitched sounds in the science of harmony and of musical composition" could accommodate these adaptations, "the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent" (1842)

20.16. Reading: Riskin: The Defecating Duck, or, the Ambiguous Origins of Artificial Life

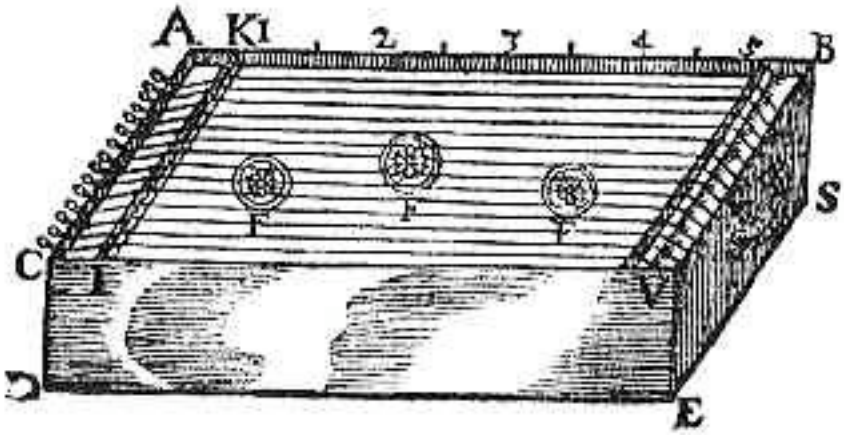
- Riskin, J. 2003. "The Defecating Duck, or, the Ambiguous Origins of Artificial Life." *Critical Inquiry* 29(4): 599-633.
- Why does Riskin isolate the Defecating Duck as important?
- "by building a machine that played the flute and another that shat, and placing them alongside each other, Vaucanson, rather than demonstrating the equivalence of art and shit as the products of mechanical processes, was testing the capacity of each, the artistic and the organic product, to distinguish the creatures that produced them from machines" (2003, p. 610)
- It seems that, even though some knew that The Turk was not truly autonomous, many still appreciated its functioning: why?
- "not only has our understanding of what constitutes intelligence changed according to what we have been able to make machines do but, simultaneously, our understanding of what machines can do has altered according to what we have taken intelligence to be"
- Is music a sign of intelligence? How does this affect our interpretation of musical automata?
- How might musical automata employ similar theatrical, or fraudulent, presentations?

20.17. Musical Automata

- Take many forms, often encoding data on a disc or cylinder
- Nondeterministic musical automata often used wind
  - The aeolian harp



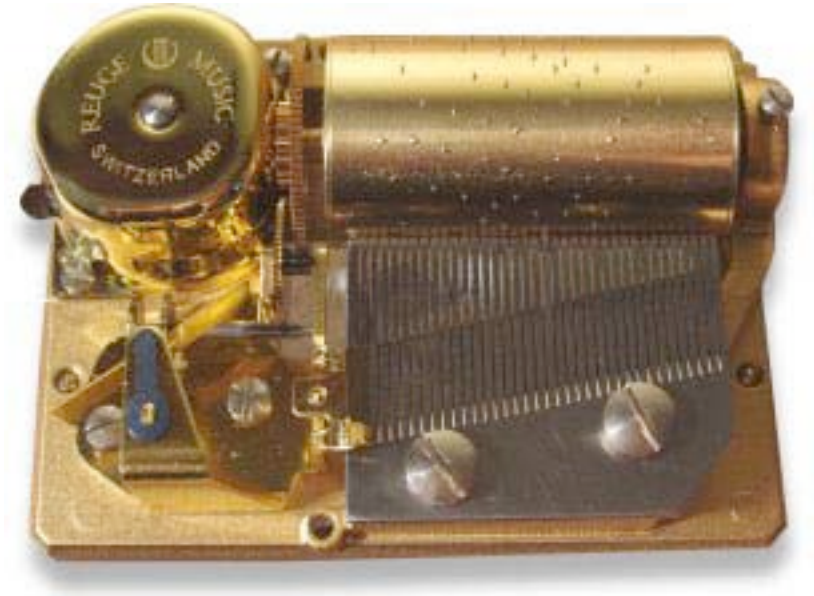
(Public domain image)



- Various arrangements of wind bells used in East and South-East Asia

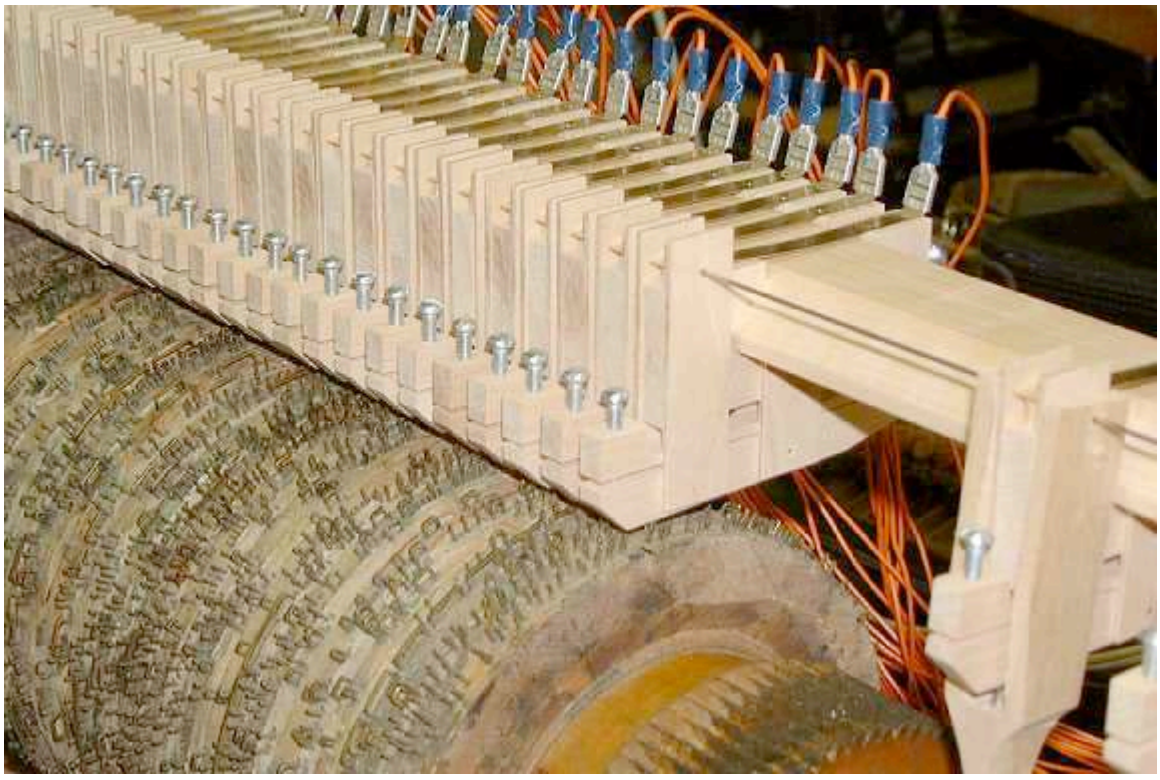
- The windchime

- Mechanical models often deterministic
  - Music boxes from the early 19th cenury



- Barrel organs employed a wide range of sounds sources

(Public domain image)



- Reproducing and player pianos were by far the most widespread musical automata

- Some player pianos encouraged human interaction and intervention in the performance

20.18. Weinberg: Haile

- Robotic percussionist that listens and performs with live players

- Developed by Gil Weinberg at Georgia Tech (2006)
- Haile videos at http://www.cc.gatech.edu/~gilwein/Haile.htm - CNN, Jam'aa Odense Outdoor

Chapter 21. Meeting 21, Workshop

21.1. Announcements

- Sonic system reports due and presentations begin: 11 May
- Last quiz: Tuesday, 4 May

21.2. Workshop: Sonic System Project Drafts

- `All student present their draft projects`

Chapter 22. Meeting 22, Approaches: Agents and Ecological Models

22.1. Announcements

- Sonic system reports due and presentations begin: 11 May
- Last quiz: Tuesday, 4 May

22.2. Workshop: Sonic System Project Drafts

- `Last two students present their draft projects`

22.3. Agents

- Software models of autonomous sub-systems
- Complexity and emergent behavior through the interaction of simple agents

22.4. Interactive Music Systems

- Computers that musically respond to MIDI messages (control data)
- Computers that musically respond to audio (sound through a microphone)
- Computers that accompany a human performance based on a shared score
- Computer (agents) that musically respond to each other (via audio or MIDI)

22.5. Analysis and Generation

- Interactive systems must have two basic components
- Components that "listen" to control data or audio information, and decode into musical models
- Components that generate musical responses based on analysis

22.6. Interactivity: Theatre

- Musical performance is theatre

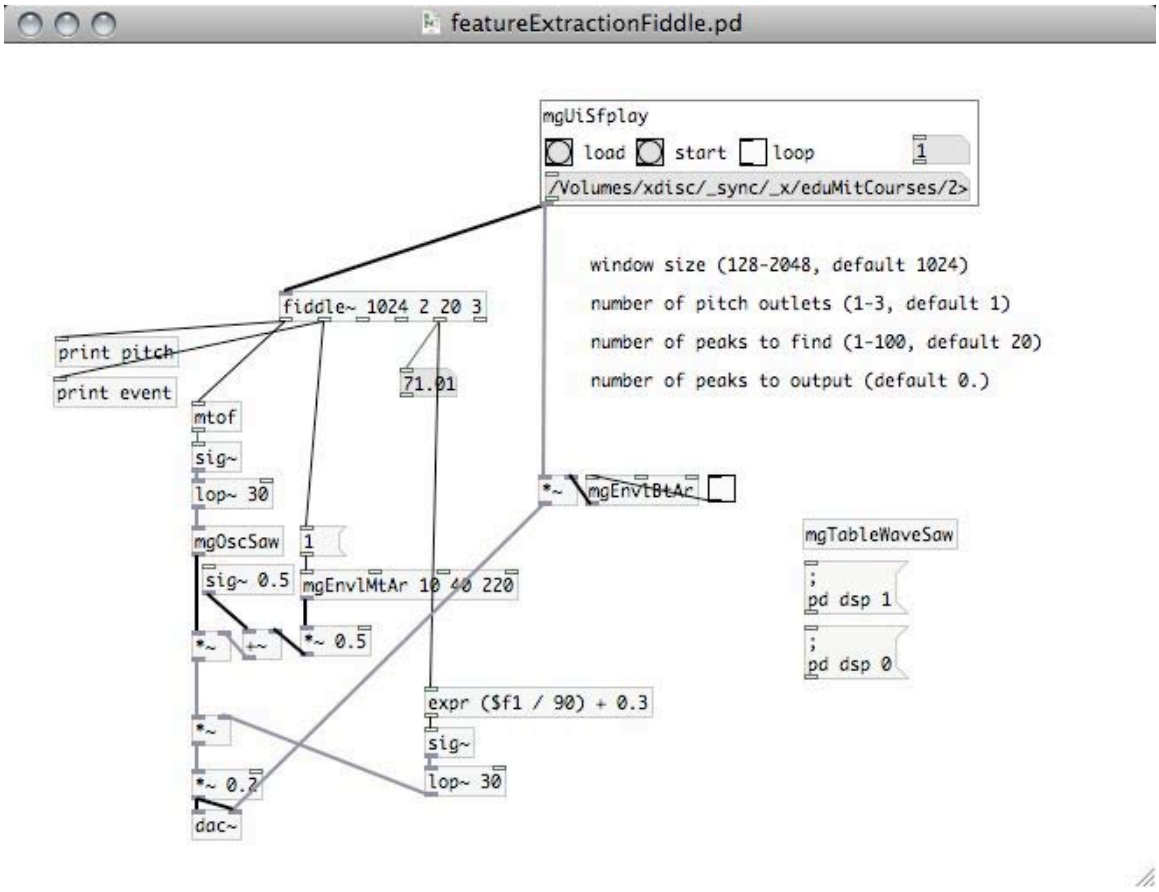
- Appeal of technological achievement or drama of technological disaster

22.7. Applications of Multi-Parameter Feature Extraction

- Detect articulation, pitch, and tempo and match to a score: score following
- Detect articulation, pitch, and rhythms, and build musical responses: interactive systems, installations

22.8. Multi-Parameter Feature Analysis in PD

- [fiddle~] object: pitch, event, and amplitude



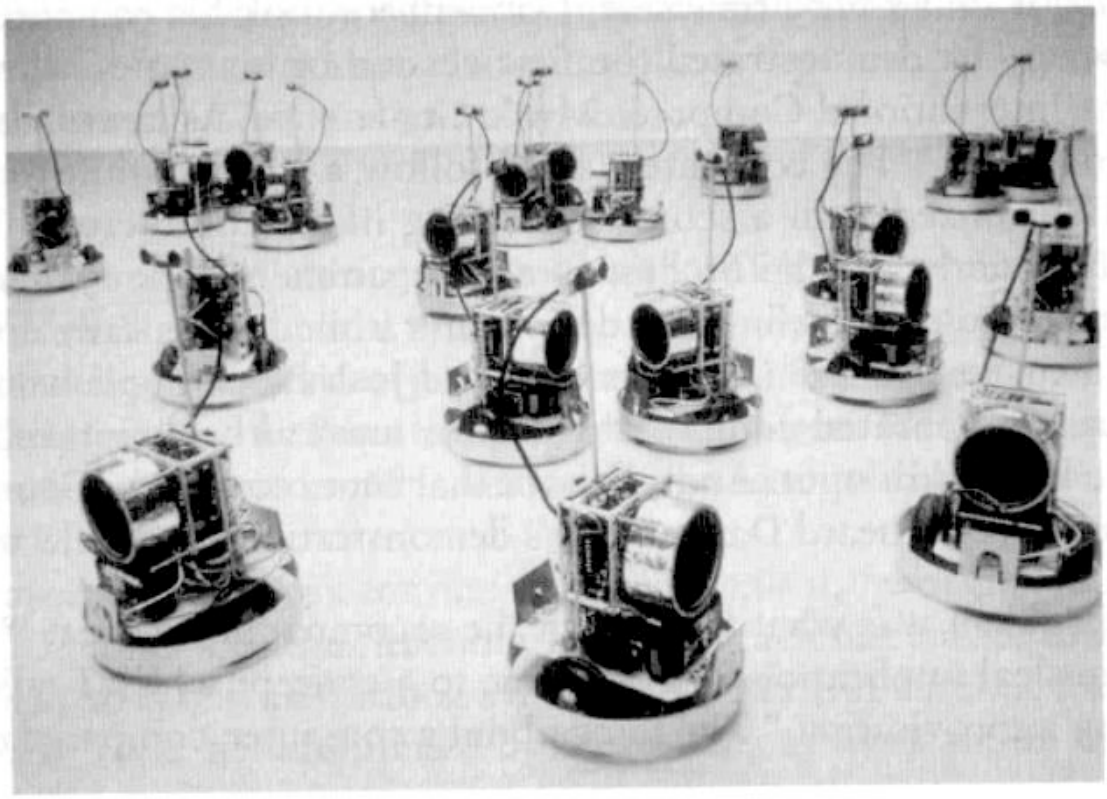
22.9. Early Historical Examples of Interactive Music Systems

- 1967: Gordon Mumma's *Hornpipe* (1967): "an interactive live electronic work for solo hornists, cybersonic console, and a performance space"; system analyzes sound form horn and in performance space

Audio: local (file://localhost/Volumes/xdisc/_sync/_x/eduMitCourses/21m380b/audio/mummaHornpipe.mp3)

- 1968: Max Mathews and F. Richard Moore develop GROOVE system at Bell Labs. Real-time performance interface to a predetermined musical score
- 1979: George Lewis, with a KIM-1 computer, develops interactive compositions designed to work with improvisation
- 1983: Felix Hess creates 40 *Electronic Sound Creatures*, small mobile machines with microphones and speakers that respond to each other an the envrionment

Felix Hess' Moving Sound Creatures in the late 1980s. Photo by John Stoel. Courtesy Felix Hess.



- 1987: Robert Rowe develops a system called Cypher, consisting of a Listener, a Player, and a Critic, used in *Flood Gate* (1989)

22.10. Reading: Rowe: Machine Listening and Composing with Cypher

- Rowe, R. 1992. "Machine Listening and Composing with Cypher." *Computer Music Journal* 16(1): 43-63.
- What types of features are extracted during the first level of listener analysis?
- What types of features are extracted during the second level of analysis?
- How does the chord and key analysis routines work?
- What are the three compositional methods employed?
- What is the role of the critic?
- How is the large-scale behavior of the system varied over time?

22.11. Listening: Rowe

- Listening: Robert Rowe, *Shells*, 1993

22.12. Listening: Ariza

- Listening: Christopher Ariza, *to leave the best untold*, 2009

22.13. Alternative Agent Models

- Analogies to human roles
- Analogies to ecological models
- Analogies to social systems
- Analogies to physical systems

22.14. A Model of Particle Feedback Systems

- Particles in a dynamic system

- Particles
  - Have one or more states, each state with a discrete life span
  - Particle expired at termination of life span
  - Life cycle:

    ```
    [('a', 1), ('b', 2)]
    ```
- Particle Transformers
  - Have one or more states, each state with a discrete life span
  - Particle expired at termination of life span
  - State determines focus of particle
  - Focus is target state looked for in other particles; transformed with transformation map
  - Transform map:

    ```
    {'a':[(None, 3), ('a', 1)]}
    ```
  - Related to first order Markov chain
- Sensor Producers
  - Produces one type of Particle
  - Produces one type of Particle Transformer
  - Stores a threshold, a target value for a given state
  - Senses the composition of a collection of Particles
  - Stores a production count range: given difference from threshold, give a range of Particles to produce (when below threshold) or Particle Transformers to produce (when above threshold).
  - Production count range:

    ```
    {(-30,-10): [1,2], (1,10): [1, 2], (11, 20): [1, 4], None: [1, 8]}
    ```
- Environment
  - Store lists of Sensor Producers, Particles, and Particle Transformers
  - Provides model of Sensor Producer (one for now)
  - Provides an absolute discrete value range for sensed particle

- Specify number of sensors
- Can age all Particles by one or more age steps

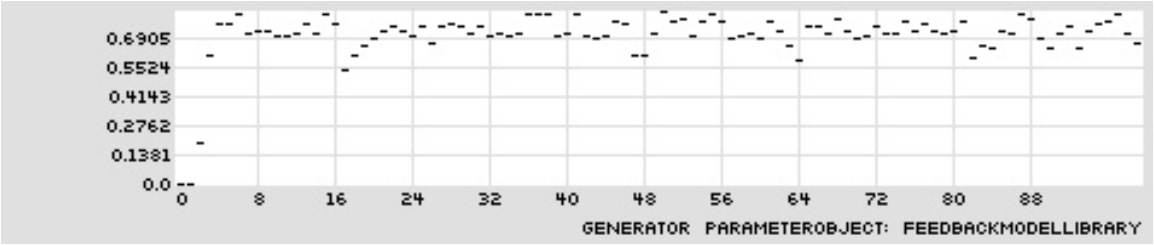
22.15. Feedback System as ParameterObject

- The feedbackModelLibrary ParameterObject

```
:: tpv fml
Generator ParameterObject
{name,documentation}
FeedbackModelLibrary feedbackModelLibrary, feedbackModelName, parameterObject,
                     parameterObject, min, max
                     Description: Produces values from a one-dimensional string
                     rewrite rule, or Lindenmayer-system generative grammar. The
                     terminus, or final result of the number of generations of
                     values specifed by the stepCount parameter, is used to
                     produce a list of defined values. Values are chosen from
                     this list using the selector specified by the
                     selectionString argument. Arguments: (1) name, (2)
                     feedbackModelName, (3) parameterObject {aging step}, (4)
                     parameterObject {threshold}, (5) min, (6) max
```

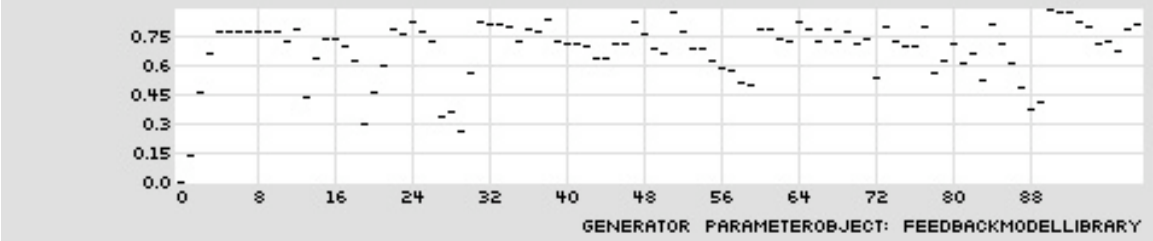
- A basic model of a Thermostat: particles as heat

```
:: tpmap 100 fml,t,(bg,rc,(1,1.5,2))
feedbackModelLibrary, thermostat, (basketGen, randomChoice, (1,1.5,2)),
(constant, 0.9), (constant, 0), (constant, 1)
TPmap display complete.
```



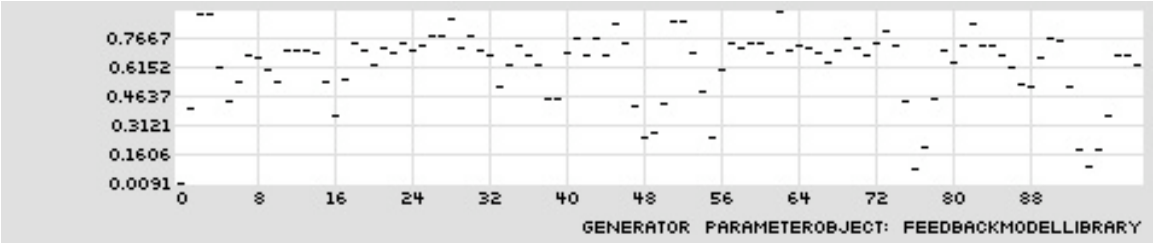
- Dynamic age values applied to Particles

```
:: tpmap 100 fml,t,(ls,e,(c,30),0,4)
feedbackModelLibrary, thermostat, (lineSegment, (constant, 30), (constant, 0),
(constant, 4)), (constant, 0.9), (constant, 0), (constant, 1)
TPmap display complete.
```



- Climate control: produce both Particles and Particle Transformers

```
:: tpmap 100 fml,cc,(bg,rc,(.5,1,1.5))
feedbackModelLibrary, climateControl, (basketGen, randomChoice, (0.5,1,1.5)),
(constant, 0.9), (constant, 0), (constant, 1)
TPmap display complete.
```



- Alternative approaches to PO interface?

22.16. Feedback System as Dynamic Contour

- Can treat the grammar alphabet as parameter values: integers, floating point values
- Command sequence:
  - emo mp
  - tmo lg
  - tin a 66
  - *constant pulse*

    tie r pt,(c,8),(c,1),(c,1)
  - *amplitude controlled by Thermostat feedback*

    tie a fml,t,(bg,rc,(1,1.5,2))
  - *using convert second to set durations*

tie r cs,(fml,t,(c,1),(c,.7),.001,.400)

- *amplitude controlled by Climate Control feedback*

  tie a fml,cc,(bg,rc,(.5,1,1.5)),(c,.7),0,1

- eln; elh

22.17. Feedback System as Path Index Values

- Feedback system states as index values from the Path
- Command sequence:
  - emo m
  - *create a single, large Multiset using a sieve*

    pin a 5@1|7@4,c2,c7

  - tmo ha
  - tin a 107
  - *constant rhythm*

    tie r pt,(c,4),(c,1),(c,1)

  - *select only Multiset 0*

    tie d0 c,0

  - *create only 1 simultaneity from each multiset; create only 1-element simultaneities*

    tie d2 c,1; tie d3 c,1

  - *select pitches from Multiset using Thermostat*

    tie d1 fml,t,(bg,rc,(1,1.5,2)),(c,.7),0,18

  - *select pitches from Multiset using Climate Control*

    tie d1 fml,cc,(bg,rc,(.5,1,1.5)),(c,.7),0,18

  - eln; elh

Chapter 23. Meeting 23, Approaches: Expert Systems and Style Emulation

23.1. Announcements

- Sonic system reports due and presentations begin: 11 May

23.2. A Model of a Particle Feedback Systems

- Review

23.3. Quiz

- 10 Minutes

23.4. Style Experts

- Style emulation
    1. One of the earliest approaches to generative music
    2. Often justified has offering musicological or analytical features
    3. Often valued because various forms of testing are possible
- Expert systems
    1. Sometimes considered a type of AI
    2. Typically used to solve non-trivial problems where probabilistic recommendations are valuable
    3. Notoriously narrow

23.5. Reading: Ebcioglu: An Expert System for Harmonizing Four-part Chorales

- Ebcioglu, K. 1988. "An Expert System for Harmonizing Four-part Chorales." *Computer Music Journal* 12(3): 43-51.
- What is meant by the term "analysis by synthesis'? Is such an approach broadly applicable in music?"
- Why do the author's reject the approach of coding musical rules in a programming language, and instead offer what?
- The authors distinguish their approach from that of generating a random solution and testing the results: what is different in their approach?
- In general terms, describe the steps used to create chorales.
- Did the authors find that published theory texts were sufficient to implement their generative techniques?
- What are the main chorale views used?
- What level of results do the author's report achieving?
- What hardware were the using, and how quickly could it solve harmonizations?

23.6. Recreating Works of the Past

- Is their historical or musicological value to recreating works of the past?
- Is their aesthetic or artistic value to recreating the works of the past?
- Listen: 21 April 2006: Radio Lab, WNYC, show #202

23.7. Reading: Cope: Computer Modeling of Musical Intelligence in EMI

- Cope, D. 1992. "Computer Modeling of Musical Intelligence in EMI." *Computer Music Journal* 16(2): 69-83.
- What are the basic steps used in the production of music with Cope's EMI system?

- How is music represented and what parameters are taken into account?
- What role does the pattern matcher play? What does it match?
- Given Cope's description, how does the augmented transition network (ATN) differ from something like a Markov chain?
- Is Cope's description of EMI sufficient to understand the techniques of production?
- Cope claims that musical intelligence is a "simulation of musical thinking"; does the EMI system approach musical intelligence?

23.8. Listening: Bach, Virtual Bach, and Cope

- Listening: Cope, *Three Inventions, after Bach, No. 1*, 1997
- Compare to Bach Invention No. 6 in E Major

- Listening: Cope, *Three Inventions, after Bach, No. 2*, 1997
- Compare to Bach Invention No. 8 in F Major

- Listening: Cope, *Three Inventions, after Bach, No. 3*, 1997

[Page left blank]

Chapter 24. Meeting 24, Discussion: Aesthetics and Evaluations

24.1. Announcements

- Sonic system reports due and presentations begin: 11 May

24.2. Quiz Review

- ?

24.3. The (Real) Turing Test

- Turing, A. M. 1950. "Computing Machinery and Intelligence." *Mind* 59: 433-460.

VOL. LIX. No. 236.] [October, 1950

MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

BY A. M. TURING

1. *The Imitation Game.*

I PROPOSE to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think'. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.

The new form of the problem can be described in terms of a game which we call the 'imitation game'. It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either 'X is A and Y is B' or 'X is B and Y is A'. The interrogator is allowed to put questions to A and B thus:

C: Will X please tell me the length of his or her hair?

Now suppose X is actually A, then A must answer. It is A's

- A test of human and computer indistinguishability

Image by MIT OpenCourseWare.

- Based on a party game in which an interrogator attempts to distinguish the gender of two human agents
- Through removing biases (sound, visual presence), and focusing on language alone, can a machine be indistinguishable from a human?
- Multiple tests can be averaged; after 5 minutes of conversation correct identification must be less than 70 percent
- Claim only of achieving thinking, not intelligence
- Functional rather than structural indistinguishability (2000, p. 429)

- Deception is permitted: mathematical questions can take longer, or fake mistakes
- Is human-like conversation the sole determinate of thinking?

24.4. The Eliza Effect

- Humans too easily associate humanity with machines
- Eliza in emacs: shift + escape; enter "xdoctor" and return

24.5. Other Tests: The John Henry Test

- The John Henry Test (JHT): a test of verifiable distinguishability between human and machine
- Other examples?

24.6. Other Tests: The Turing Hierarchy

- Steven Harnad
- Total Turing Test: full physical and sense based interaction
- T4: internal microfunctional indistinguishability
- T5: microphysical indistinguishability, real biological molecules
- t1: toy tests: subtotal fragments of our functional capacity (Harnad 2000, p. 429)
- The TT is predicated on total functional indistinguishability; anything less is a toy

24.7. A Little Turing Test

- Hofstadter, D. R. 1979. *Gödel, Escher, Bach: an eternal golden braid* . New York: Vintage.
- The little turing test (1979, p. 621)

A Little Turing Test

Below, I have reproduced nine selections, carefully culled from many pages of output from later versions of my program. Along with them are three (seriously intended) human-written sentences. Which?

(1) Blurting may be considered as the reciprocal substitution of semiotic material (dubbing) for a semiotic dialogical product in a dynamic reflexion.

(2) Rather think of a pathway of a 'sequence' of gedankenexperiment simpletons where heir-lines are a prima facie case of a paradiachronic transitivity.

(3) Think of that as a chain strength possibility of what, eventually, comes out as a product (epistemic conditions?) and the product is not a Frankfurt-ish packing-it-all-in.

(4) Despite the efforts, the reply, if you will, had been supported by the Orient; hence a fallacy will thereafter be suspended by the attitude which will be being held by the ambassador.

(5) Of course, until the upheavals, the ambassador was slightly gradually mollycoddling the rabble.

(6) Supposedly, refined liberty caused the attitudes insofar as peace is distilled by the consequences which will not eventually be caused by the command irrevocably insofar as peace of it is sometimes causing the intransigency infinitesimally surprisingly.

(7) According to the sophists, the campaigns in the city-states, in other words, have been accepted by the Orient cunningly. Of course, the Orient has been separated by the states particularly violently.
The Orient supports the efforts which had been supported by mankind.

(8) Admittedly, the hierarchical origin of the fallacy, nevertheless, will be prophesied by the enemies of it. By the same token, the individualists will have testified that intransigency will not have suspended the campaigns.

(9) Needless to say, during the upheaval which will have warranted the secrecy, the replies do not separate the Orient. Of course, the countries, ipso facto, are always probing liberty.

(10) Although a Nobel Prize was being achieved by the humanists, yet in addition, it was being achieved by the serf.

(11) An attitude will often be held by the serfs of a strife-torn nation.

(12) Moreover, the Nobel Prizes will be achieved. By the same token, despite the consequence, the Nobel Prizes which will be achieved will sometimes be achieved by a woman.

- Is this a Turing Test?

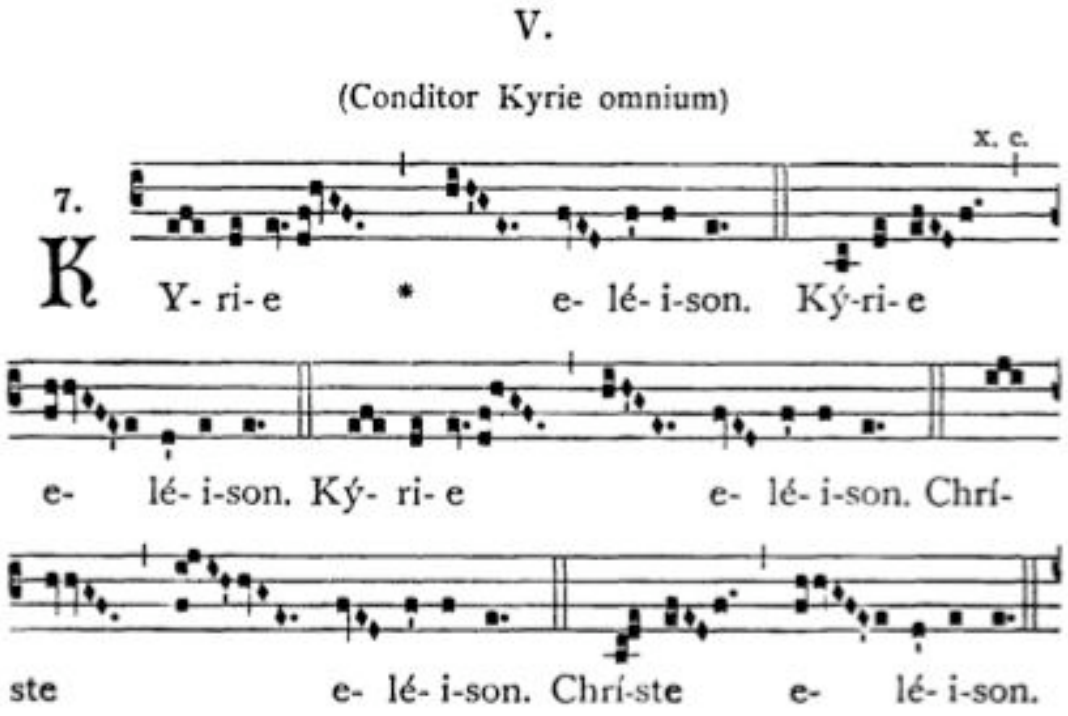
24.8. A (Kind of) Turing Test

- Kurzweil, R. 1990. *The Age of Intelligent Machines*. Cambridge: MIT Press.
- "The essence of the Turing Test is that the computer attempts to act like a human within the context of an interview over terminal lines. A narrower concept of a Turing test is for a computer to successfully imitate a human within a particular domain of human intelligence. We might call these domain-specific Turing tests. One such domain-specific Turing test, based on a computer's ability to write poetry, is presented here." (1990, p. 374)
- 28 question "poetic Turing test" administered to 16 human judges; 48 percent correct overall
- Cybernetic Poet

  http://www.kurzweilcyberart.com/poetry/rkcp_akindofturingtest.php
- "Music composed by computer is becoming increasingly successful in passing the Turing test of believability. The era of computer success in a wide range of domain-specific Turing tests is arriving." (1990, p. 378)
- Kurzweil and Kapor Long Bet: 20,000 that a machine will pass the Turing Test by 2029
- Is there a narrower concept of a Turing Test?

24.9. A Musical Turing Test

- Compare chants created by computer and by humans



- Is this a Turing Test?
- How would this test be different if the music was performed by humans?

24.10. Musical Turing Test Archetypes

- Musical Directive Toy Test (MDtT)
- Musical Output Toy Test (MOtT)
- The problem of musical judgements
- Music is not natural language
- We have aesthetic expectations for human and computer music
- All executed tests report a win for the computer
- Does success of a MDtT or a MOtT offer a sign of system design success?
- Does aesthetic success suggest system design success?

24.11. Discrimination Tests

- Blind comparison of musical outputs
- Often material used to create the music is used as part of the test
- All listening test are bound by musical judgements

24.12. Cope's MOtTs

- Cope does not associate these test directly with the TT
- Compares EMI generated Mozart with Mozart
- 1992 AAAI conference conducted a test with 2000 visitors, claiming "absolutely no scientific value" but claims that "machine-composed music has some stylistic validity"
- Compares virtual music to real music in The Game
- Many have used Cope's music or related tests as examples of musical TTs where the machine wins

24.13. Machine Authorship in Generative Music Systems

- Is the machine responsible for the musical output?
- Is the test testing the machine at all?

24.14. Aesthetic Intention in Generative Music Systems

- The intentional fallacy: the idea that understanding the artist's intention is necessary for evaluating a work (Beardsley 1946)
- Is intention required to make music?
- Can authorship be given to things that do not have intention?

24.15. Listening

- Listening: David Soldier, "The Birth of Ganesha," *Elephonic Rhapsodies*, 2004

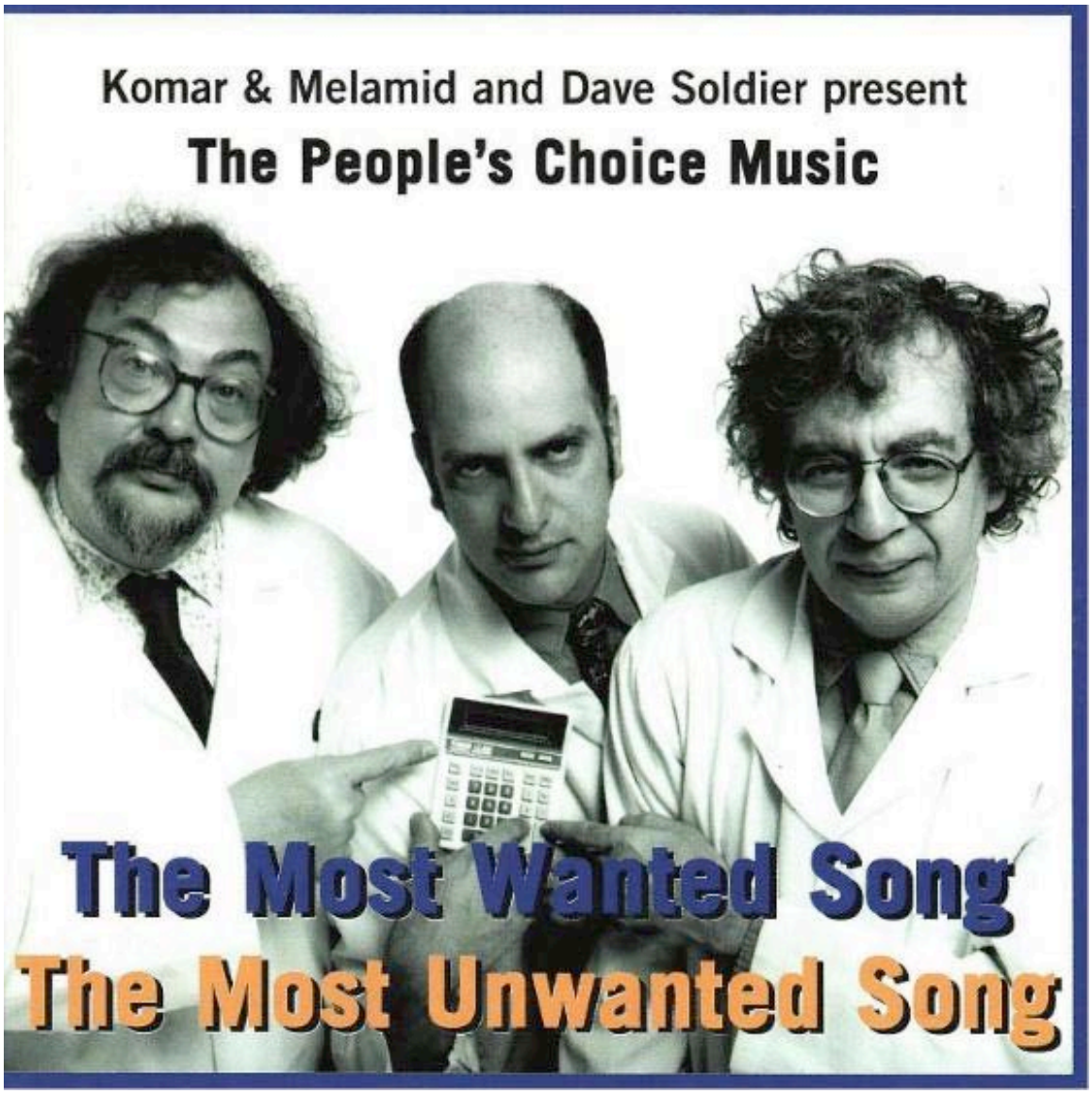
- Elephants trained and directed in improvisation with instruments

24.16. Naughtmusik

- Soldier, D. 2002. "Eine Kleine Naughtmusik: How Nefarious Nonartists Cleverly Imitate Music." *Leonardo Music Journal* 12: 53-58.
- Genuine music requires composers with intent
- Naughtmusik: nonart sounds, composers without intent
- An Adapted Turing Test: can human judges detect naughtmusik?
- The Tangerine Awkestra: children 2 to 9, produce sounds using instruments they do not know how to play, recorded in a studio; listened to free jazz of Ornette Coleman and others
- 5 sophisticated adults: 5 of 8 trials led to correct identification: not iron-clad
- Thai elephant orchestra
- "There is something out there that looks, sounds, feels, smells like music, but isn't" (2002, p. 58)

24.17. Listening

- The People's Choice Music: with Vitaly Komar and Alex Melamid
- Survey given to 500 Americans
- Survey responders had no intent; the works were created without individual intent, and thus no creative decision making was involved
- Listening: David Soldier, *The Peoples Choice*, 2002



Courtesy of Dave Soldier. Used with permission.

24.18. Authorship Matters

- Humans are still ultimately responsible for machine creations
- The designation of author is a special designation, granted only by humans

- Authorship does not require intention: what does it require?

Chapter 25. Meeting 25

25.1. Announcements

- Sonic system reports are due tonight by midnight

25.2. A Negative Definition

- A CAAC system is software that facilitates the generation of new music by means other than the manipulation of a direct music representation (Ariza 2005b)

25.3. Sonic System Presentations

- `Half of the class presents today, half at next class`

Chapter 26. Meeting 26

26.1. Announcements

•

26.2. Sonic System Presentations

- 2nd half of the class presents today

References

Ames, C. 1987. "Automated Composition in Retrospect: 1956-1986." *Leonardo* 20(2): 169-185.

Ames, C. 1989. "The Markov Process as a Compositional Model: A Survey and Tutorial." *Leonardo* 22(2): 175-187.

Ames, C. 1991. "A Catalog of Statistical Distributions: Techniques for Transforming Random, Determinate and Chaotic Sequences." *Leonardo Music Journal* 1(1): 55-70.

Ames, C. 1992. "A Catalog of Sequence Generators: Accounting for Proximity, Pattern, Exclusion, Balance and/or Randomness." *Leonardo Music Journal* 2(1): 55-72.

Anders, T. and E. R. Miranda. 2009. "Interfacing Manual and Machine Composition." *Contemporary Music Review* 28(2): 133-147.

Ariza, C. 2005a. *An Open Design for Computer-Aided Algorithmic Music Composition: athenaCL*. Ph.D. Dissertation, New York University.

Ariza, C. 2005b. "Navigating the Landscape of Computer-Aided Algorithmic Composition Systems: A Definition, Seven Descriptors, and a Lexicon of Systems and Research." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 765-772.

Ariza, C. 2005c. "The Xenakis Sieve as Object: A New Model and a Complete Implementation." *Computer Music Journal* 29(2): 40-60.

Ariza, C. 2006. "Beyond the Transition Matrix: A Language-Independent, String-Based Input Notation for Incomplete, Multiple-Order, Static Markov Transition Values." Internet: http://www.flexatone.net/docs/btmimosmtv.pdf.

Ariza, C. 2007a. "Automata Bending: Applications of Dynamic Mutation and Dynamic Rules in Modular One-Dimensional Cellular Automata." *Computer Music Journal* 31(1): 29-49.

Ariza, C. 2007b. "Serial RSS Sound Installation as Open Work: The babelcast." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 1: 275-278.

Ariza, C. 2009a. "The Interrogator as Critic: The Turing Test and the Evaluation of Generative Music Systems." *Computer Music Journal* 33(2): 48-70.

Ariza, C. 2009b. "Pure Data Object Glossary." Internet: http://flexatone.net/docs/pdg.

Ariza, C. 2010. "Two Experiments in the Early History of Computer-Aided Algorithmic Composition."

Assayag, G. and C. Rueda, M. Laurson, C. Agon, O. Delerue. 1999. "Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic." *Computer Music Journal* 23(3): 59-72.

Babbitt, M. 1958. "Who Cares if you Listen." *High Fidelity* 8(2): 38.

Bel, B. 1998. "Migrating Musical Concepts: An Overview of the Bol Processor." *Computer Music Journal* 22(2): 56-64.

Ben-Tal, O. and J. Berger. 2004. "Creative Aspects of Sonification." *Leonardo Music Journal* 37(3): 229-232.

Berg, P. and R. Rowe, D. Theriault. 1980. "SSP and Sound Description." *Computer Music Journal* 4(1): 25-35.

Berg, P. 1996. "Abstracting the Future: The Search for Musical Constructs." *Computer Music Journal* 20(3): 24-27.

Berg, P. 2003. *Using the AC Toolbox*. Den Haag: Institute of Sonology, Royal Conservatory.

Berg, P. 2009. "Composing Sound Structures with Rules." *Contemporary Music Review* 28(1): 75-87.

Beyls, P. 1989. "The Musical Universe of Cellular Automata." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 34-41.

Boulanger, R. C. 2000. *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming*. Cambridge: MIT Press.

Burt, W. 1996. "Some Parentheses Around Algorithmic Composition." *Organised Sound* 1(3): 167-172.

Chadabe, J. 1997. *Electric Sound: The Past and Promise of Electronic Music*. New Jersey: Prentice-Hall.

Chareyron, J. 1988. "Sound Synthesis and Processing by Means of Linear Cellular Automata." *Unpublished poster presented at the International Computer Music Conference*.

Chareyron, J. 1990. "Digital Synthesis of Self-Modifying Waveforms by Means of Linear Automata." *Computer Music Journal* 14(4): 25-41.

Chernoff, J. M. 1979. *African Rhythm and African Sensibility*. Chicago: University of Chicago Press.

Childs, E. 2002. "Achorripsis: A Sonification of Probability Distributions." *Proceedings of the 2002 International Conference on Auditory Display*.

Collins, N. 2006. *Towards Autonomous Agents for Live Computer Music: Realtime Machine Listening and Interactive Music Systems*. Ph.D. thesis, University of Cambridge.

Collins, N. 2008. "Infno: Generating Synth Pop and Electronic Dance Music On Demand." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association.

Collins, N. 2009. "Musical Form and Algorithmic Composition." *Contemporary Music Review* 28(1): 103-114.

Cope, D. 1992. "Computer Modeling of Musical Intelligence in EMI." *Computer Music Journal* 16(2): 69-83.

Cope, D. 1996. *Experiments in Musical Intelligence*. Madison, WI: A-R Editions.

Cope, D. 2000. *The Algorithmic Composer*. Madison, WI: A-R Editions.

Cope, D. 2001. *Virtual Music: Computer Synthesis of Musical Style*. Cambridge: MIT Press.

Cope, D. 2004. "A Musical Learning Algorithm." *Computer Music Journal* 28(3): 12-27.

Cope, D. 2005. *Computer Models of Musical Creativity*. Cambridge: MIT Press.

Doornbusch, P. 2002. "Composers Views on Mapping in Algorithmic Composition." *Organised Sound* 7(2): 145-156.

Ebcioglu, K. 1988. "An Expert System for Harmonizing Four-part Chorales." *Computer Music Journal* 12(3): 43-51.

Eco, U. 1989. *The Open Work*. Translated by A. Cancogni. Cambridge: Harvard University Press.

Farbood, M. and H. Kaufman, K. Jennings. 2007. "Composing with Hyperscore: An Intuitive Interface for Visualizing Musical Structure." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 2: 111-117.

Gardner, M. 1974. "Mathematical Games: The Arts as Combinatorial Mathematics, or, How to Compose Like Mozart with Dice." *Scientific American* 231(6): 132-136.

Harnad, S. 2000. "Minds, Machines and Turing." *Journal of Logic, Language and Information* 9(4): 425-445.

Hedges, S. A. 1978. "Dice Music in the Eighteenth Century." *Music and Letters* 180-187.

Hiller, L. 1956. "Abstracts: Some Structural Principles of Computer Music." *Journal of the American Musicological Society* 9(3): 247-248.

Hiller, L. 1970. "Music Composed with Computers: An Historical Survey." In *The Computer and Music*. H. B. Lincoln, ed. Ithaca: Cornell University Press. 42-96.

Hiller, L. 1981. "Composing with Computers: A Progress Report." *Computer Music Journal* 5(4): 7-21.

Hiller, L. and L. Isaacson. 1959. *Experimental Music*. New York: McGraw-Hill.

Hoffman, P. 2000. "A New GENDYN Program." *Computer Music Journal* 24(2): 31-38.

Hoffman, P. 2002. "Towards an 'Automated Art': Algorithmic Processes in Xenakis' Compositions." *Contemporary Music Review* 21(2-3): 121-131.

Hofstadter, D. R. 1979. *Gödel, Escher, Bach: an eternal golden braid* . New York: Vintage.

Koenig, G. M. 1968. "Remarks on Composition Theory."

Koenig, G. M. 1970a. "Project One." In *Electronic Music Report*. Utrecht: Institute of Sonology. 2: 32-46.

Koenig, G. M. 1970b. "Project Two - A Programme for Musical Composition." In *Electronic Music Report*. Utrecht: Institute of Sonology. 3.

Koenig, G. M. 1971. "The Use of Computer Programs in Creating Music." In *Music and Technology (Proceedings of the Stockholm Meeting organized by UNESCO)*. Paris: La Revue Musicale. 93-115.

Koenig, G. M. 1983. "Aesthetic Integration of Computer-Composed Scores." *Computer Music Journal* 7(4): 27-32.

Koenig, G. M. 1991. "Working with 'Project One': My Experiences with Computer Composition." *Interface* 20(3-4): 175-180.

Kurzweil, R. 1990. *The Age of Intelligent Machines*. Cambridge: MIT Press.

Laske, O. 1973. "In Search of a Generative Grammar for Music." *Perspectives of New Music* 12(1): 351-378.

Lovelace, A. 1842. "Translator's notes to an article on Babbage's Analytical Engine." In *Scientific memoirs: selected from the transactions of foreign academies of science and learned societies, and from foreign journals*. R. Taylor, ed. London: printed by Richard and John E. Taylor. 3: 691-731.

Luque, S. 2006. *Stochastic Synthesis: Origins and Extensions*. Masters Thesis, Institute of Sonology.

Magnus, C. 2004. "Evolving electroacoustic music: the application of genetic algorithms to time-domain waveforms." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 173-176.

Manousakis, S. 2006. *Musical L-Systems*. Masters Thesis, Institute of Sonology.

Marino, G. and M. Serra, J. Raczinski. 1993. "The UPIC System: Origins and Innovations." *Perspectives of New Music* 31(1): 258-269.

McCartney, J. 1996. "SuperCollider: a New Real Time Synthesis Language." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association.

McCartney, J. 1998. "Continued Evolution of the SuperCollider Real Time Synthesis Environment." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association.

McCracken, D. 1955. "Monte Carlo Method." *Scientific American* 192(5): 90-96.

Miranda, E. R. 1995. "Granular Synthesis of Sounds by Means of a Cellular Automaton." *Leonardo* 28(4): 297-300.

Miranda, E. R. 2000. *Composing Music With Computers*. Burlington: Focal Press.

Miranda, E. R. 2002. "Emergent Sound Repertoires in Virtual Societies." *Computer Music Journal* 26(2): 77-90.

Miranda, E. R. 2003. "On the Music of Emergent Behavior: What Can Evolutionary Computation Bring to the Musician?." *Leonardo* 36(1): 55-59.

Mozart, W. A. 1793. *Anleitung zum Componiren von Walzern so viele man will vermittlest zweier Würfel ohne etwas von der Musik oder Composition zu verstehen*. Berlin: Juhan Julius Hummel.

Olson, H. F. and H. Belar. 1961. "Aid to Music Composition Employing a Random Probability System." *Journal of the Acoustical Society of America* 33(9): 1163-1170.

Pinkerton, R. C. 1956. "Information Theory and Melody." *Scientific American* 194(2): 77-86.

Prusinkiewicz, P. and A. Lindenmayer. 1990. *The Algorithmic Beauty of Plants (The Virtual Laboratory)*. London: Springer Verlag.

Puckette, M. 1985. "A real-time music performance system." *MIT Experimental Music Studio*.

Puckette, M. 1988. "The Patcher." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 420-429.

Puckette, M. 1997. "Pure Data." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 224-227.

Puckette, M. 2002. "Max at 17." *Computer Music Journal* 26(4): 31-43.

Riskin, J. 2003. "The Defecating Duck, or, the Ambiguous Origins of Artificial Life." *Critical Inquiry* 29(4): 599-633.

Roads, C. 1979. "Grammars as Representations for Music." *Computer Music Journal* 3(1): 48-55.

Roads, C. 1980. "Interview with Max Mathews." *Computer Music Journal* 4(4): 15-22.

Roads, C. 1988. "Introduction to Granular Synthesis." *Computer Music Journal* 12(2): 11-13.

Rowe, R. 1992. "Machine Listening and Composing with Cypher." *Computer Music Journal* 16(1): 43-63.

Schillinger, J. 1941. *The Schillinger System of Musical Composition*. New York: Carl Fischer.

Schillinger, J. 1948. *The Mathematical Basis of the Arts*. New York: Carl Fischer.

Serra, M. 1993. "Stochastic Composition and Stochastic Timbre: GENDY3 by Iannis Xenakis." *Perspectives of New Music* 31(1): 236-257.

Soldier, D. 2002. "Eine Kleine Naughtmusik: How Nefarious Nonartists Cleverly Imitate Music." *Leonardo Music Journal* 12: 53-58.

Sowa, J. F. 1957. "A machine to compose music." In *Geniac Manual*. New York: Oliver Garfield Company.

Standage, T. 2002. *The Turk*. New York: Walker & Company.

Standage, T. 2003. "Monster in a Box." Wired. Internet: http://www.wired.com/wired/archive/10.03/turk_pr.html.

Sturm, B. L. 2006. "Adaptive Concatenative Sound Synthesis and Its Application to Micromontage Composition." *Computer Music Journal* 30(4): 46-66.

Taube, H. 1997. "An Introduction to Common Music." *Computer Music Journal* 21(1): 29-34.

Taube, H. 2004. *Notes from the Metalevel: An Introduction to Computer Composition*. Amsterdam: Swets & Zeitlinger Publishing.

Tipei, S. 1989. "Manifold Compositions: A (Super)Computer-Assisted Composition Experiment in Progress." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 324-327.

Truax, B. 1985. "The PODX System: Interactive Compositional Software for the DMX-1000." *Computer Music Journal* 9(1): 29-38.

Vercoe, B. 1986. *CSOUND: A Manual for the Audio Processing System and Supporting Programs*. Cambridge: MIT Media Lab.

Voss, R. F. and J. Clarke. 1978. "1/f Noise in Music: Music from 1/f Noise." *Journal of the Acoustical Society of America* 63(1): 258-263.

Weinberg, G. and S. Driscoll. 2006. "Toward Robotic Musicianship." *Computer Music Journal* 30(4): 28-45.

Wimsatt, W. K. and M. C. Beardsley. 1946. "The Intentional Fallacy." *Sewanee Review* 54: 468-488.

Winkler, T. 1998. *Composing Interactive Music*. Cambridge: MIT Press.

Xenakis, I. 1955. "La crise de la musique sèrielle." *Gravesaner Blätter* 1.

Xenakis, I. 1960. "Elements of Stochastic Music." *Gravesaner Blätter* 18: 84-105.

Xenakis, I. 1965. "Free Stochastic Music from the Computer. Programme of Stochastic music in Fortran." *Gravesaner Blätter* 26.

Xenakis, I. 1971. "Free stochastic Music." In *Cybernetics, art and ideas*. J. Reichardt, ed. Greenwich: New York Graphic Society. 124-142.

Xenakis, I. 1985. "Music Composition Treks." In *Composers and the Computer*. C. Roads, ed. Los Altos: William Kaufmann, Inc.

Xenakis, I. 1987. "Xenakis on Xenakis." *Perspectives of New Music* 25(1-2): 16-63.

Xenakis, I. 1990. "Sieves." *Perspectives of New Music* 28(1): 58-78.

Xenakis, I. 1992. *Formalized Music: Thought and Mathematics in Music*. Indiana: Indiana University Press.

Xenakis, I. 1996. "Determinacy and Indeterminacy." *Organised Sound* 1(3): 143-155.

Zicarelli, D. 1987. "M and Jam Factory." *Computer Music Journal* 11(4): 13-29.

21M.380 Music and Technology: Algorithmic and Generative Music
Spring 2010