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:

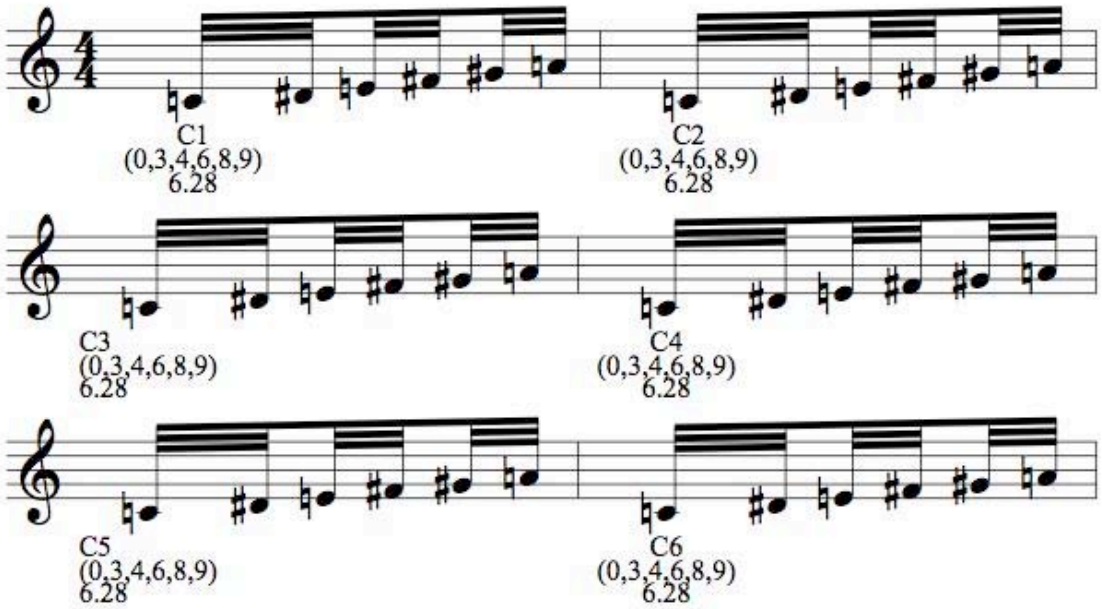
```
PIn b 5@3|7@2,c2,c4,c2,.5  5@1|7@8,c3,c6,c2,.5
```

13.6. Avoiding Octave Redundancy

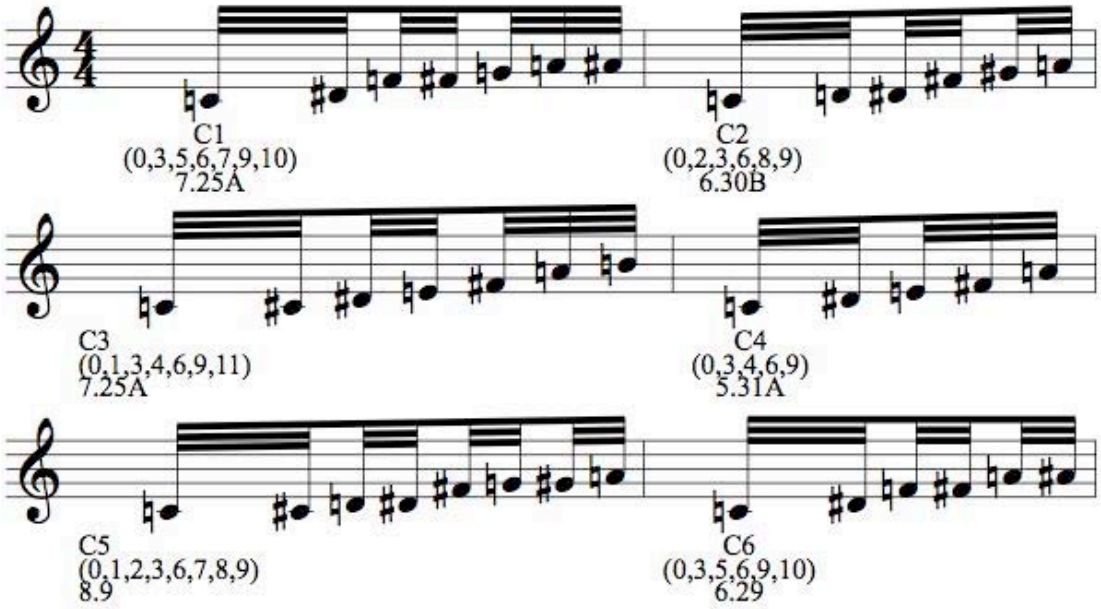
- Pitch sieves with large periods (or not a divisor or multiple of 12) are desirable
- Can be achieved simply through the union of two or more moduli with a high LCMs

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

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

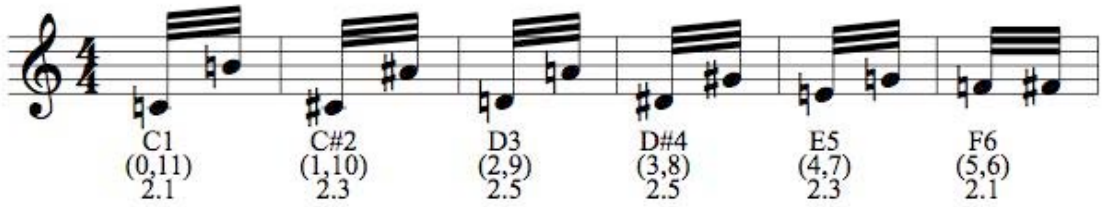



```
>>> a = sieve.Sieve('3@0|5@0|7@0')
>>> a.period()
105
```

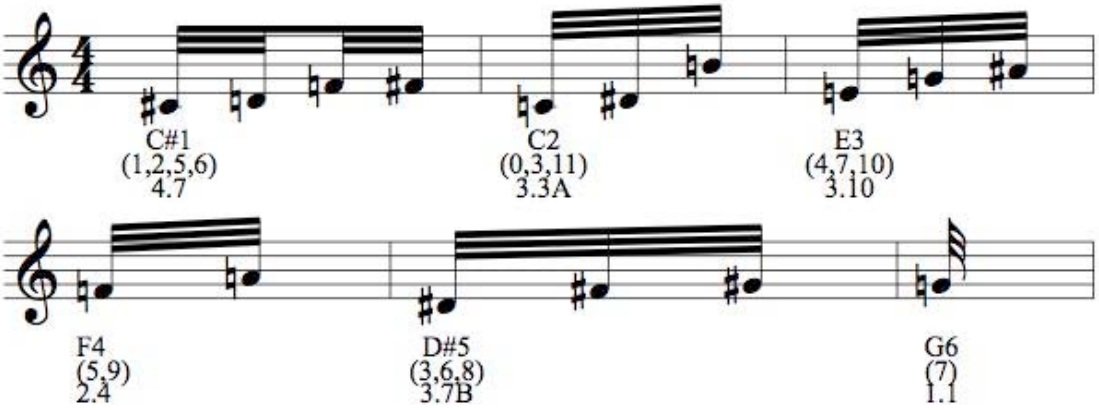


- Can be achieved through the use of moduli deviating from octave multiples (11, 13, 23, 25, 35, 37)

```
>>> 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

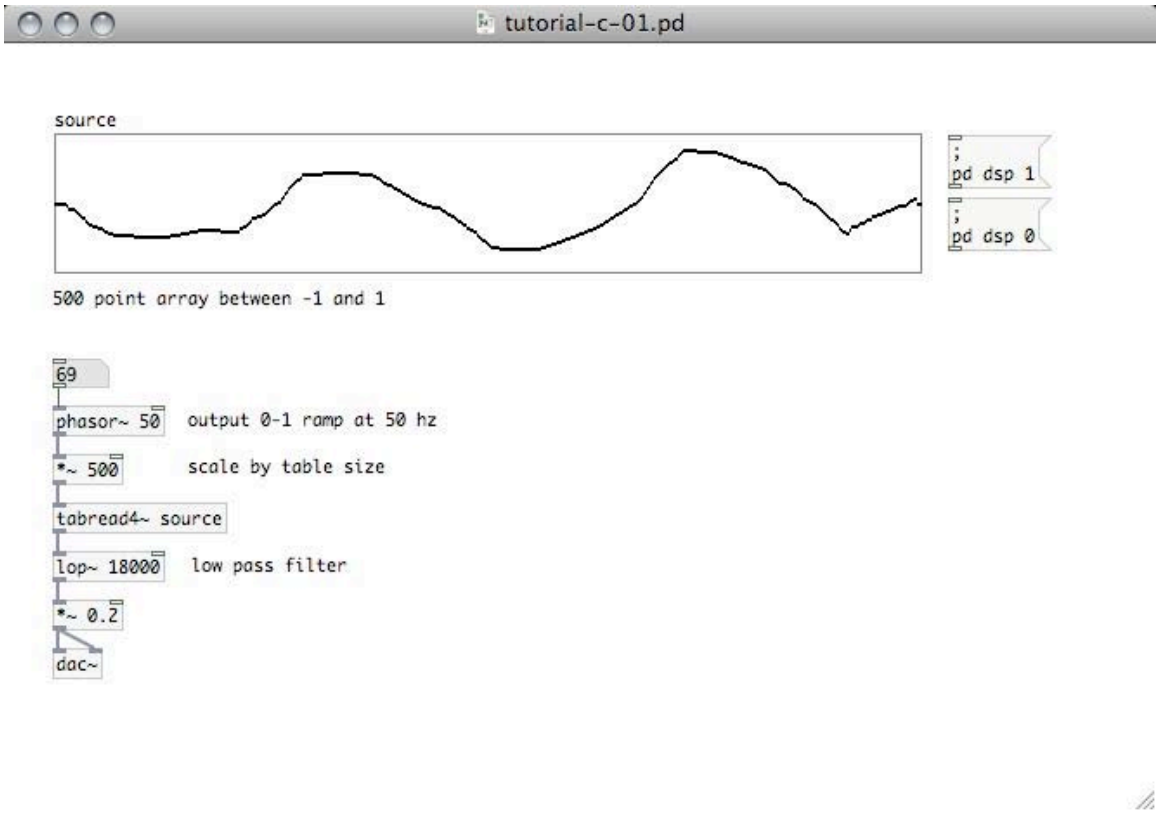
13.9. Non-Standard Synthesis: Xenakis and Koenig

- 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

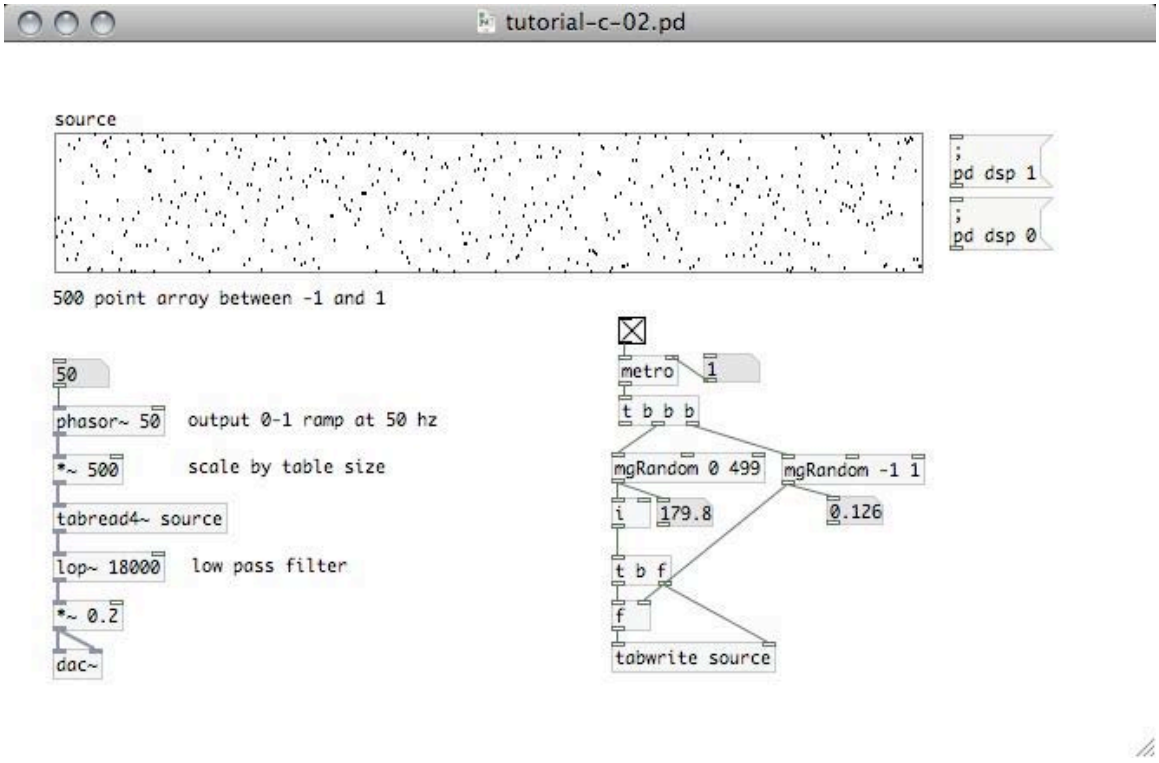
  [tabread4~] interpolates between points for any [phasor~] rate



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

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

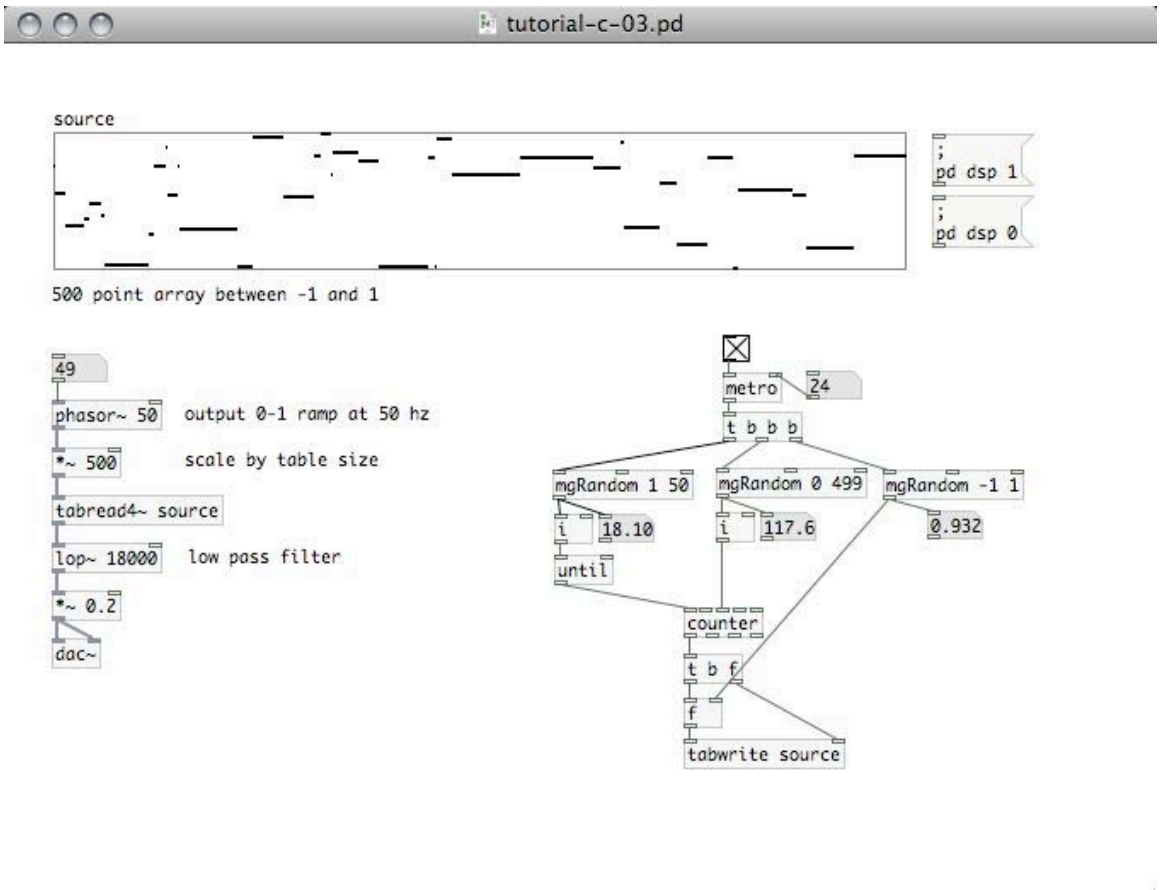
  Can only be done at the event rate (1 ms updates)



- 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

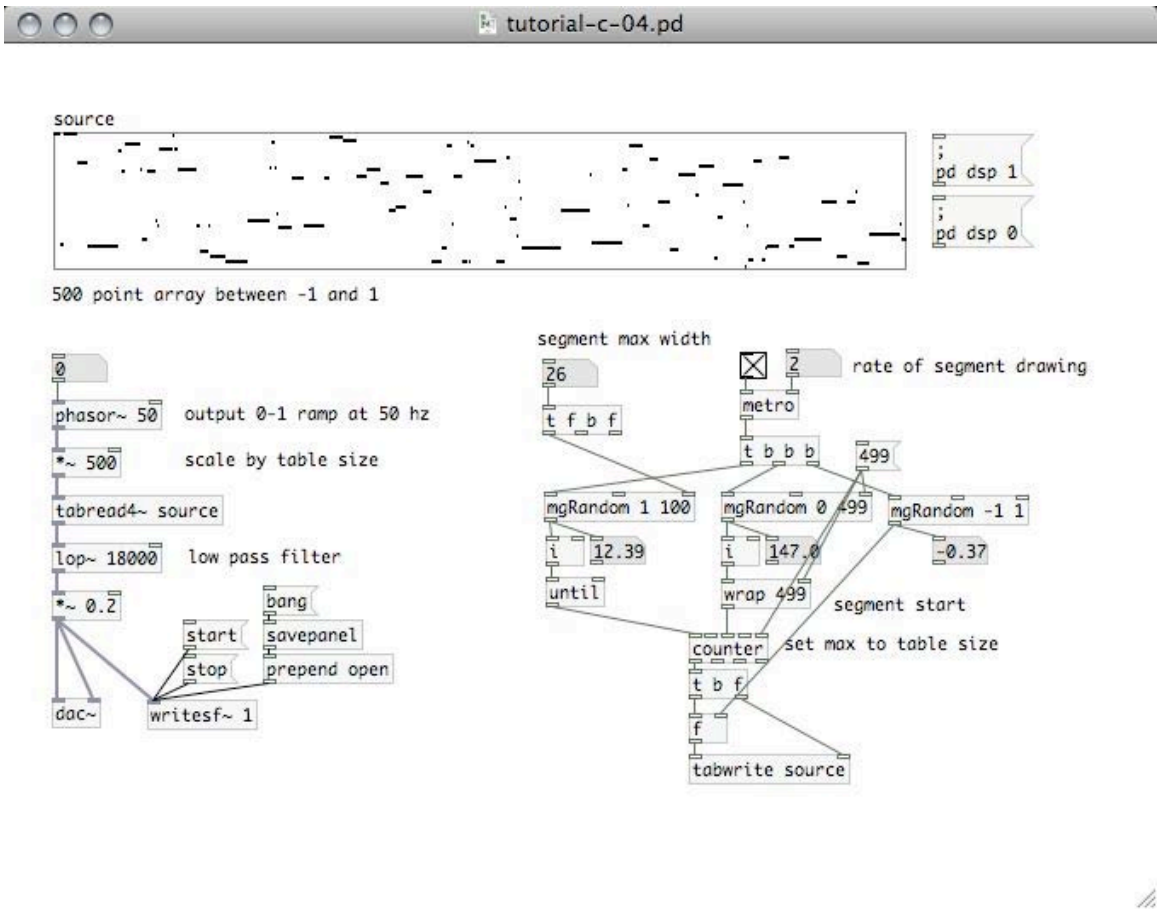


- Randomly draw line segments instead of points

  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

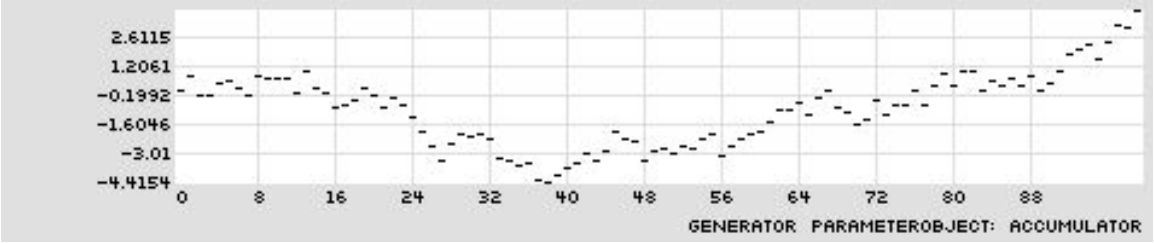
13.14. Reading: Hoffman. A New GENDYN Porgram

- 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

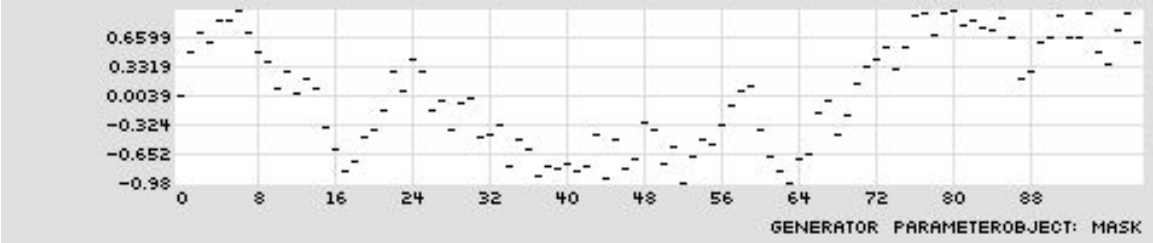
- Accumulator: permit consecutively summing values based on an initial value and the output of a ParameterObject

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



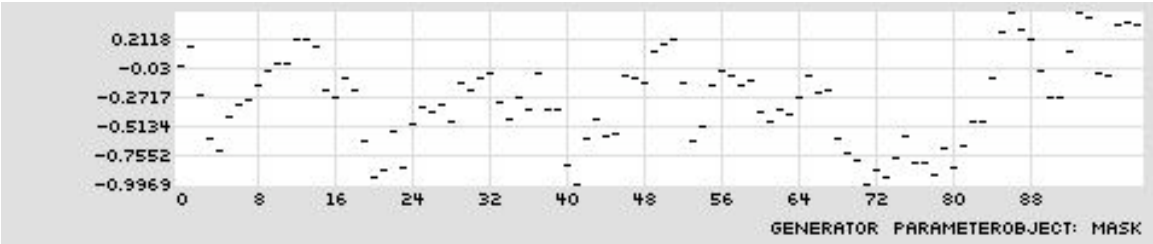
- Mask:Êconstrain the output of a ParameterObject within boundaries created by two ParameterObjects; boundaries can be limit, wrap, or reflect

```
:: 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.
```



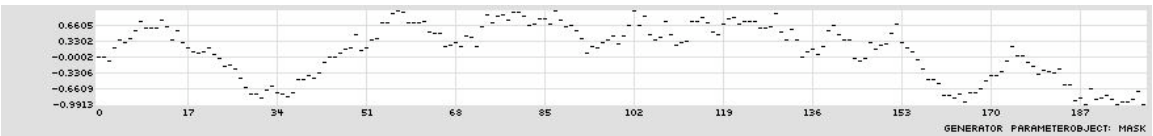
- Second order random walk: use (discrete) random walks to control the step size of another random walk

```
:: 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.
```

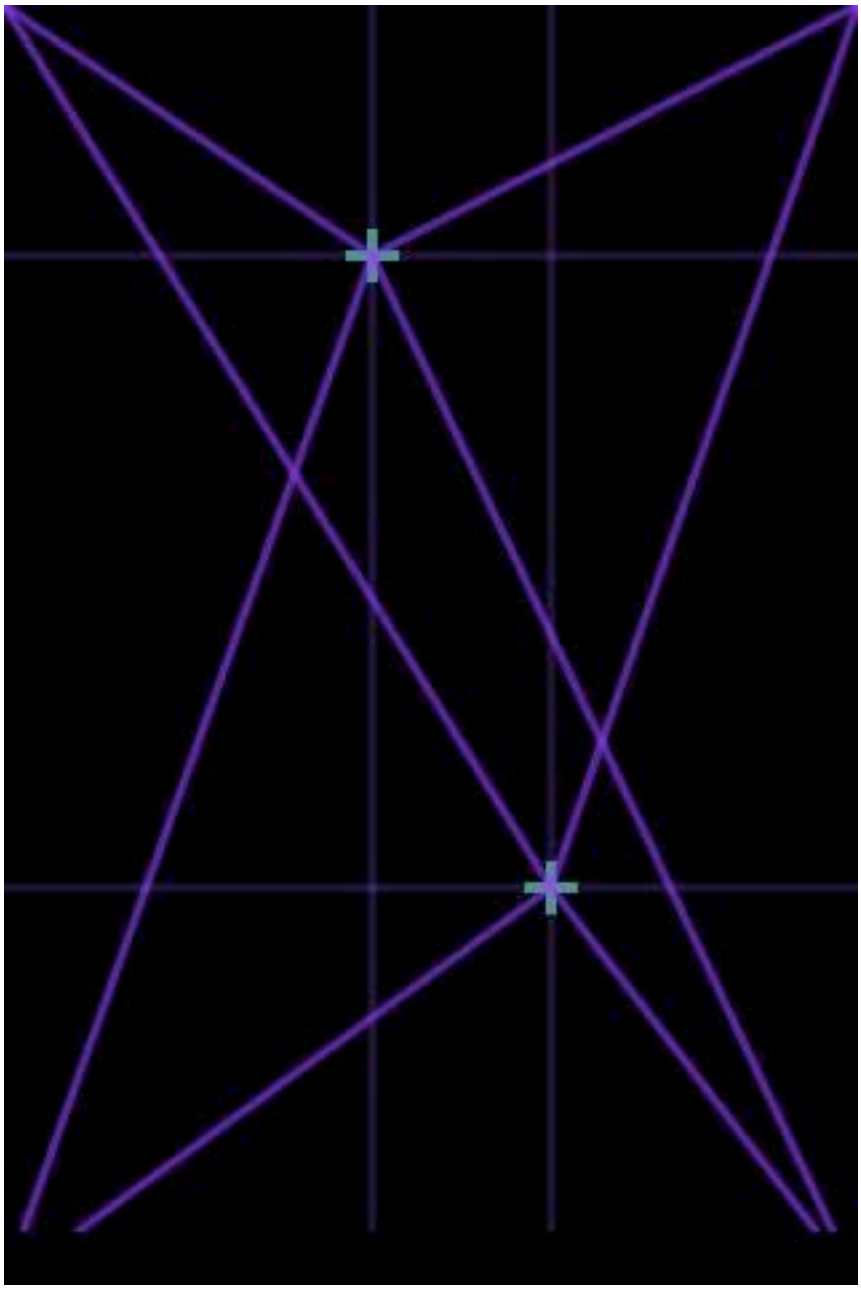


13.16. Listening: Xenakis

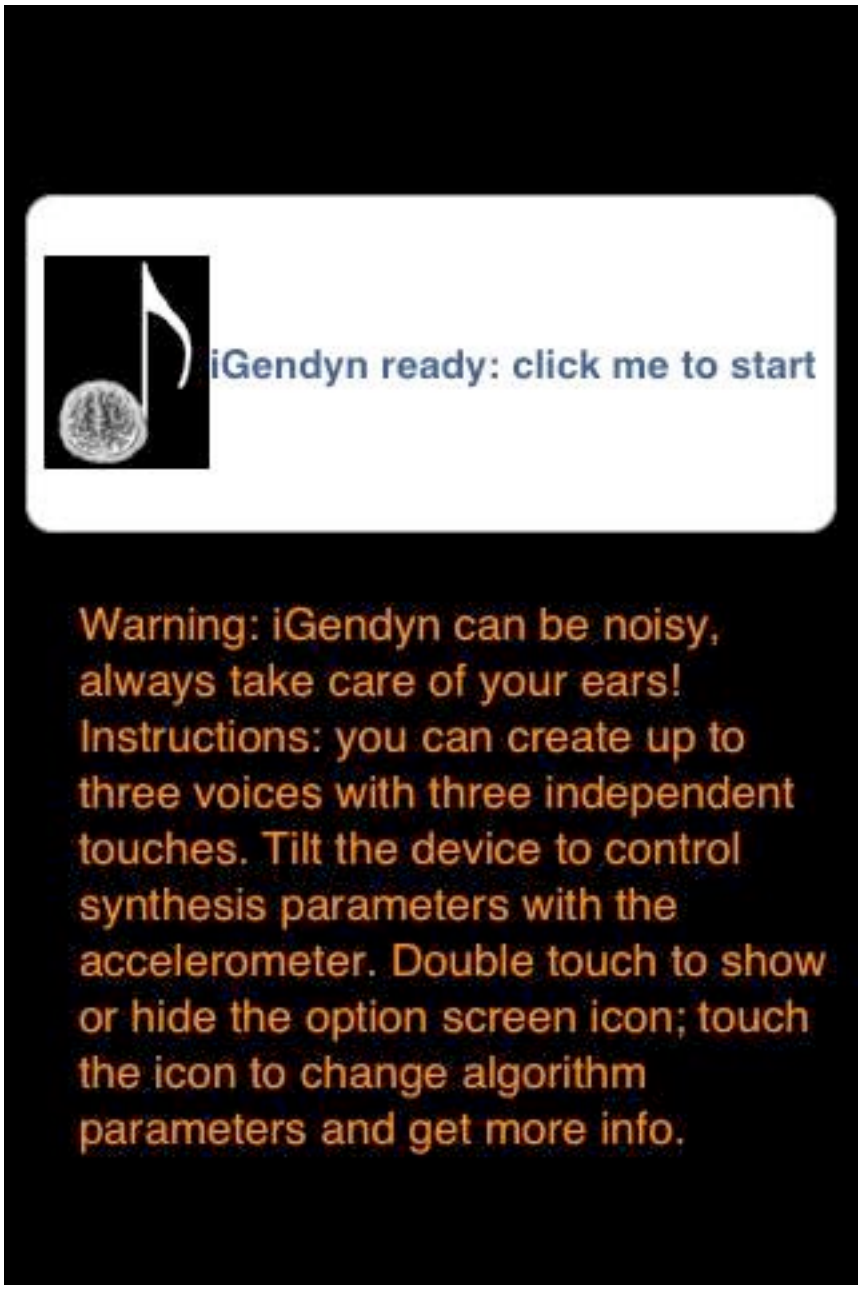
- Audio: S.709
- BBC interview with Xenakis on S.709
- "Music is not a language. Every musical piece is like a highly complex rock with ridges and designs engraved wtihin and without, that can be interpreted in a thousand ways without a single one being the best or the most true." (Xenakis 1987, p. 32)

13.17. iGendyn: Gendyn as iPhone / iPod touch App

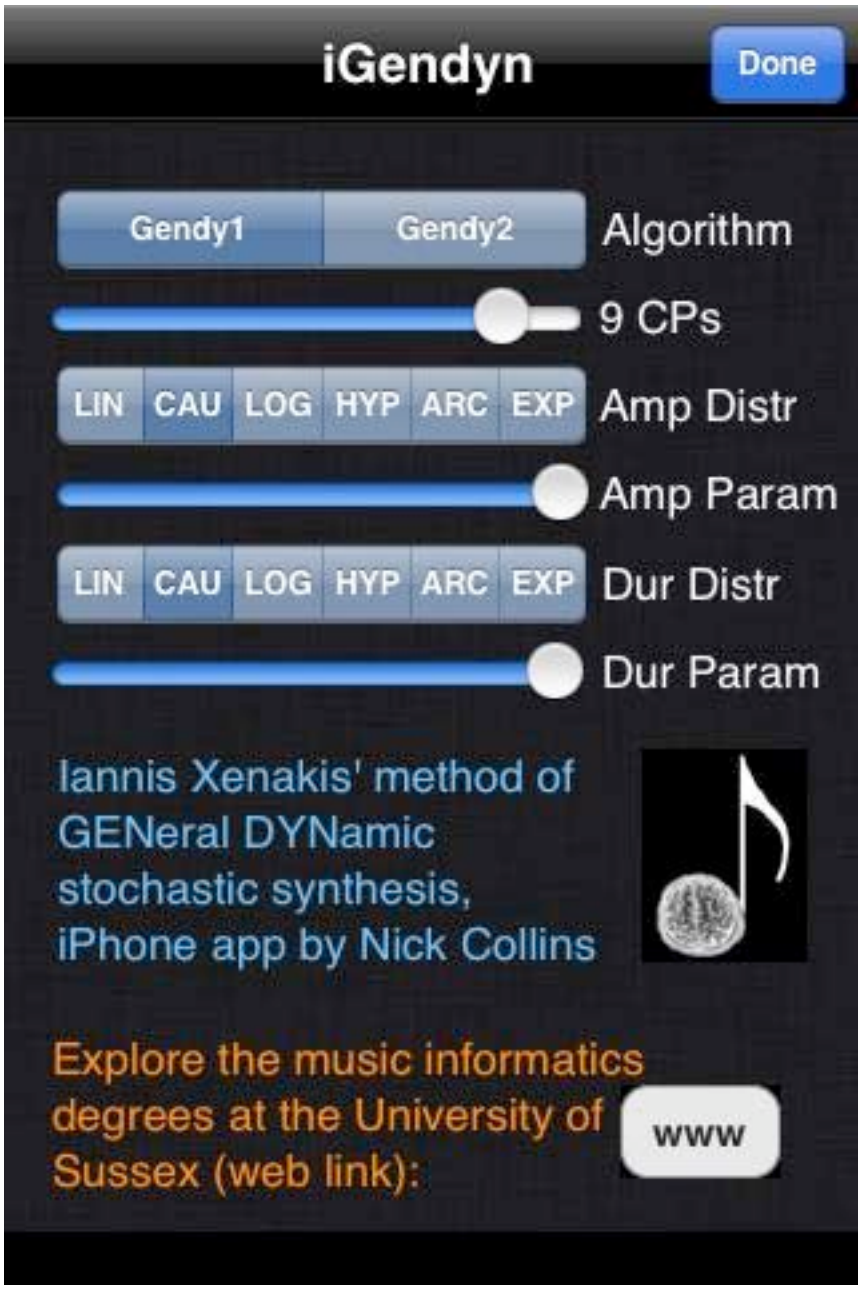
- Created by Nick Collins



Courtesy of Nick Collins. Used with permission.



Courtesy of Nick Collins. Used with permission.



Courtesy of Nick Collins. Used with permission.

21M.380 Music and Technology: Algorithmic and Generative Music
Spring 2010