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

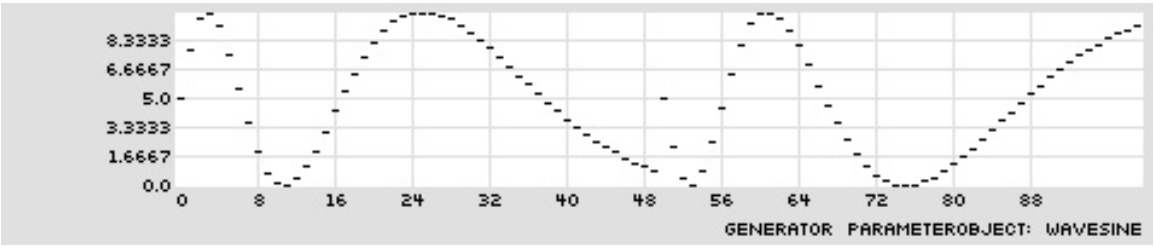
10.1. Announcements

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

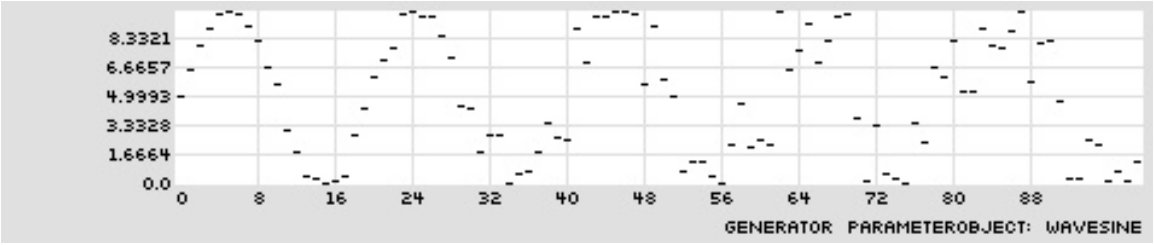
10.2. Half-Period Oscillators as ParameterObjects

- 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.
```



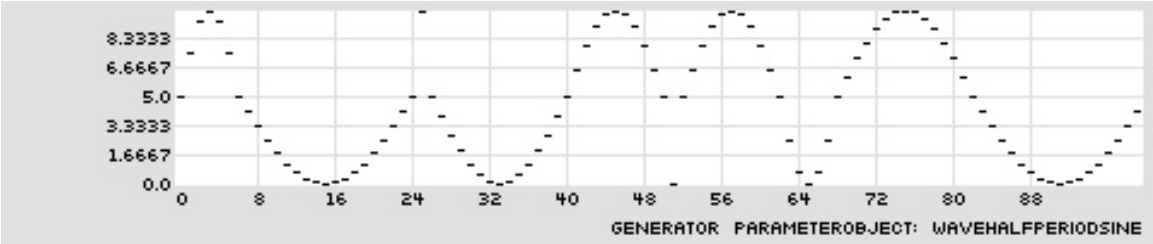

```
:: tpmap 100 ws,e,(ru,19,21),0,0,10
waveSine, event, (randomUniform, (constant, 19), (constant, 21)), 0, (constant,
0), (constant, 10)
TPmap display complete.
```



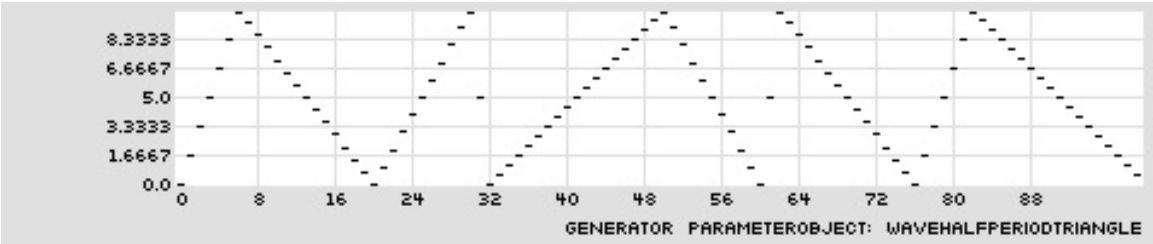
- An alternative is an oscillator that only updates seconds per half cycle (half frequency) once per half-period

WaveHalfPeriodSine, WaveHalfPeriodTriangle, WaveHalfPeriodPulse, WaveHalfPeriodCosine

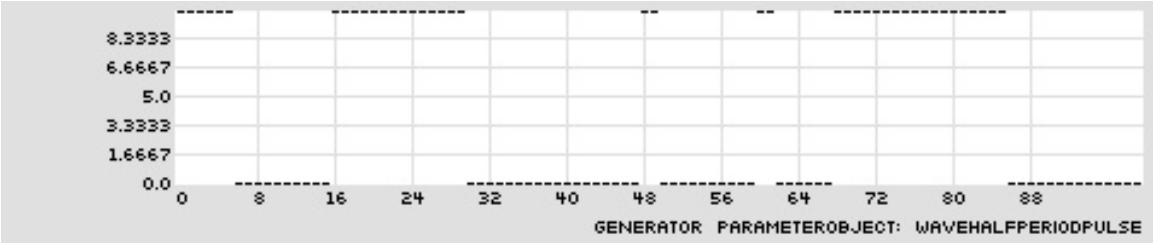
```
:: tpmap 100 whps,e,(bg,rp,(2,6,10,14,18)),0,0,10
waveHalfPeriodSine, event, (basketGen, randomPermutate, (2,6,10,14,18)), 0,
(constant, 0), (constant, 10)
TPmap display complete.
```




```
:: 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

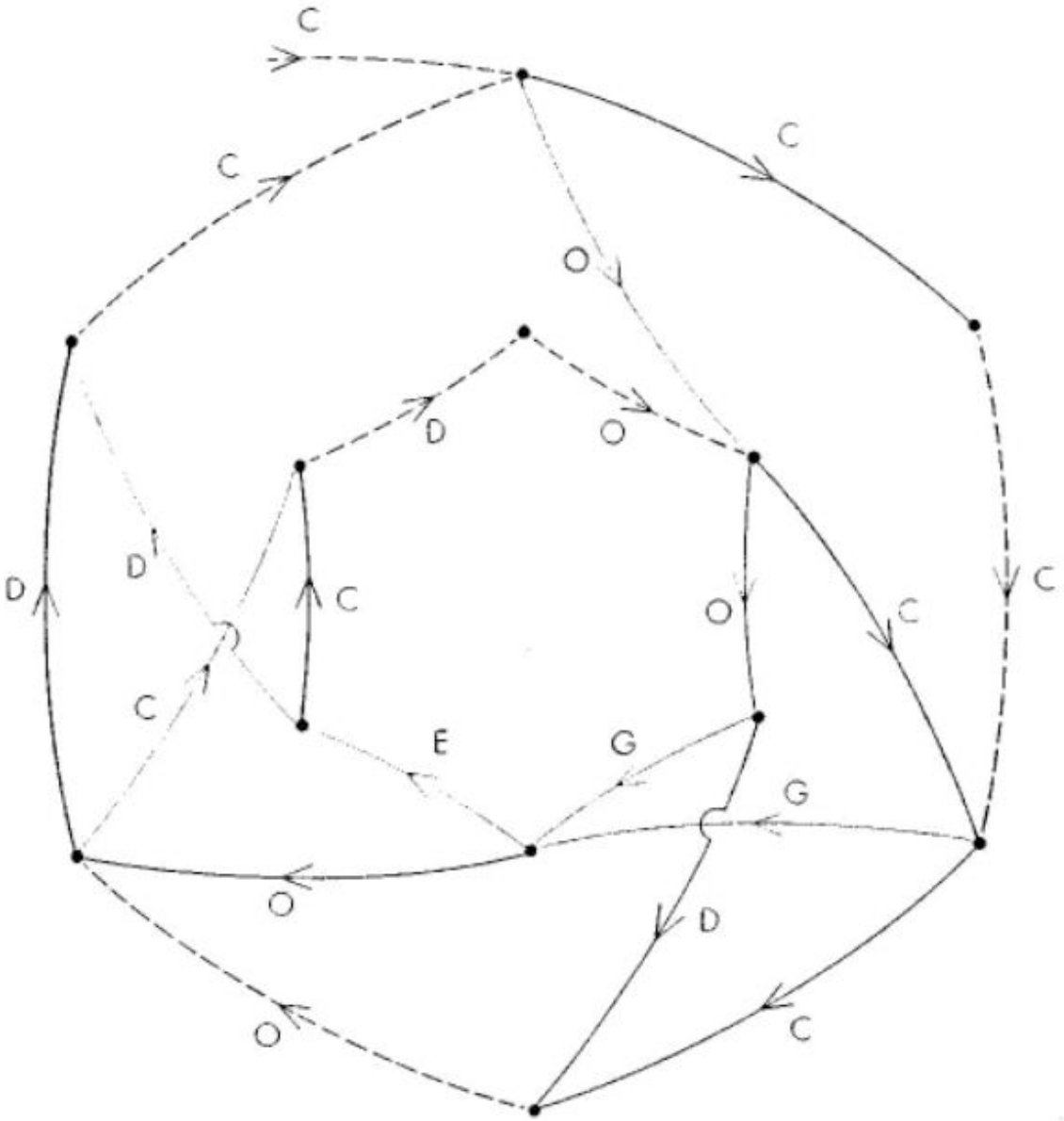
- 1906: Andrey Andreyevich Markov, Russian mathematician

  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."

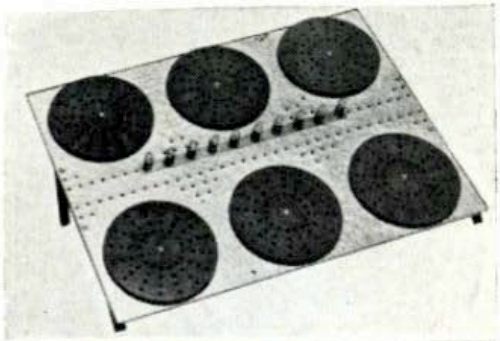
Oliver Garfield Co., Inc. Dept. ASF-108

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.

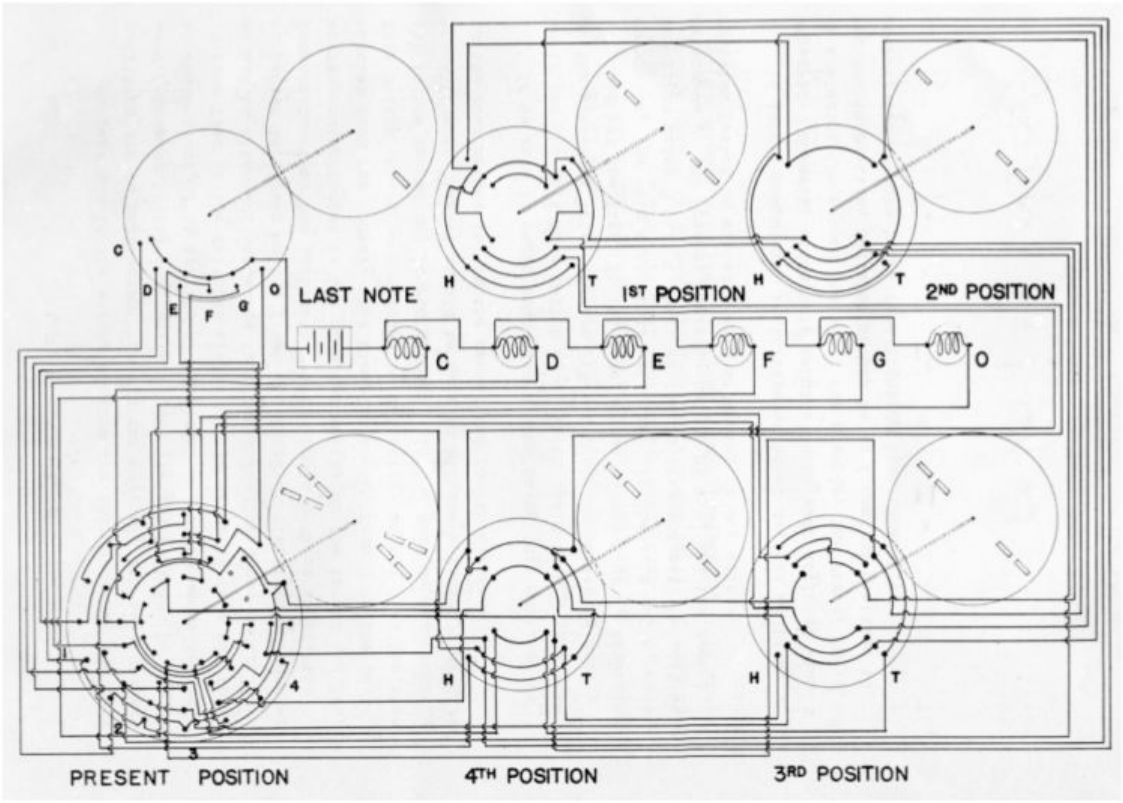
COMMENTS BY CUSTOMERS

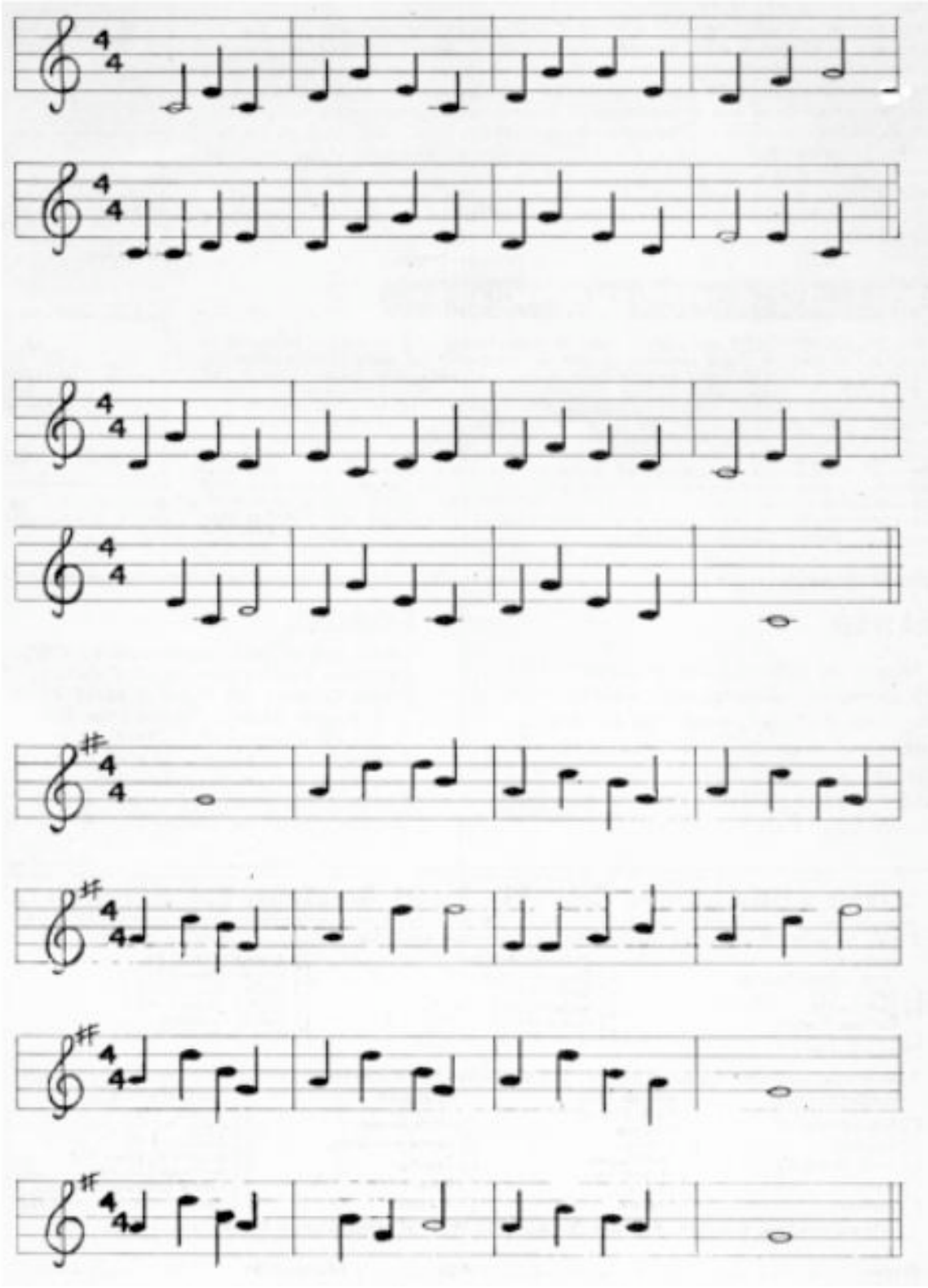
"Several months ago I purchased your **GENIAC** *Kit and found it an excellent piece of equipment. I learned a lot about computers from the enclosed books and pamphlets and I am now designing a small relay computer which will include arithmetical and logical units . . . another of my pet projects in 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.

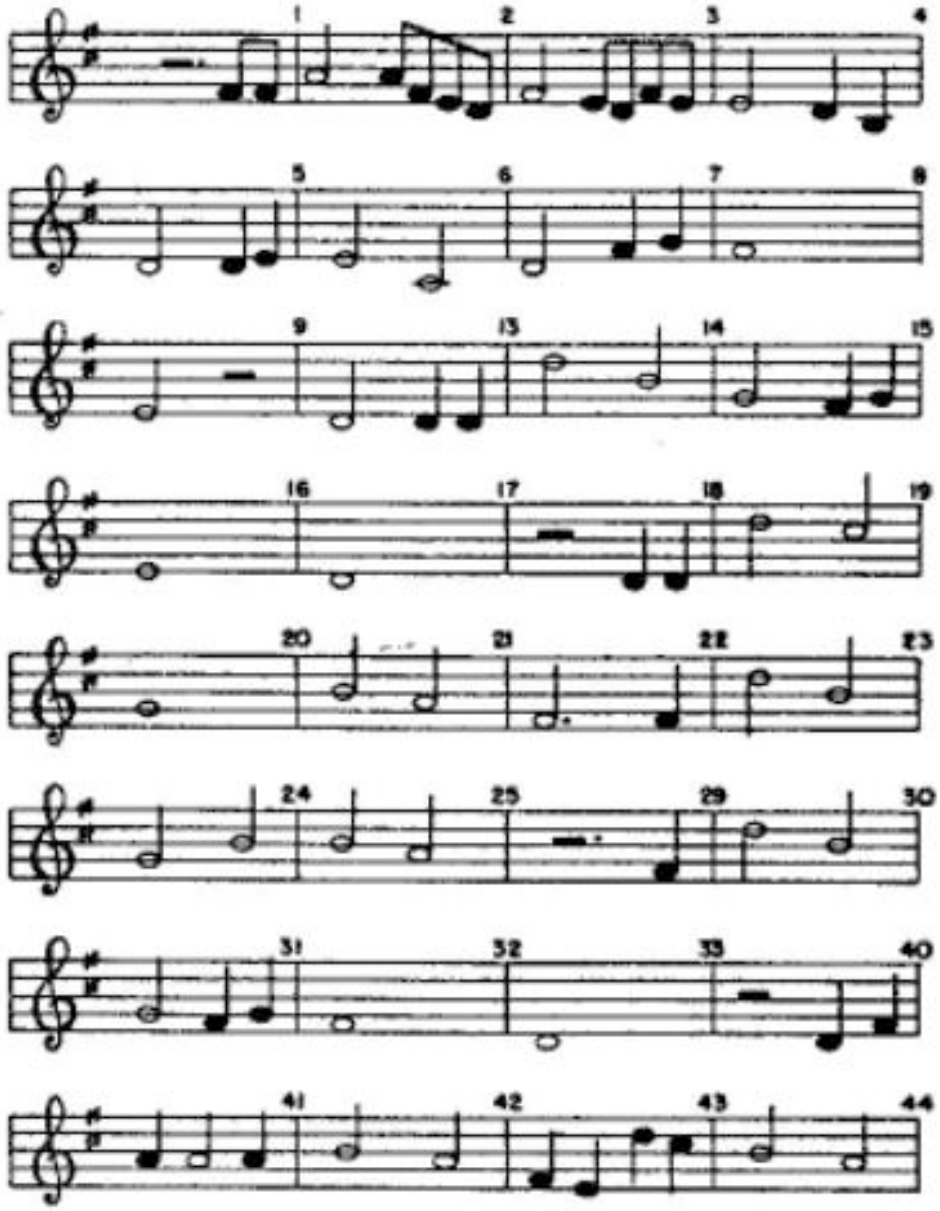
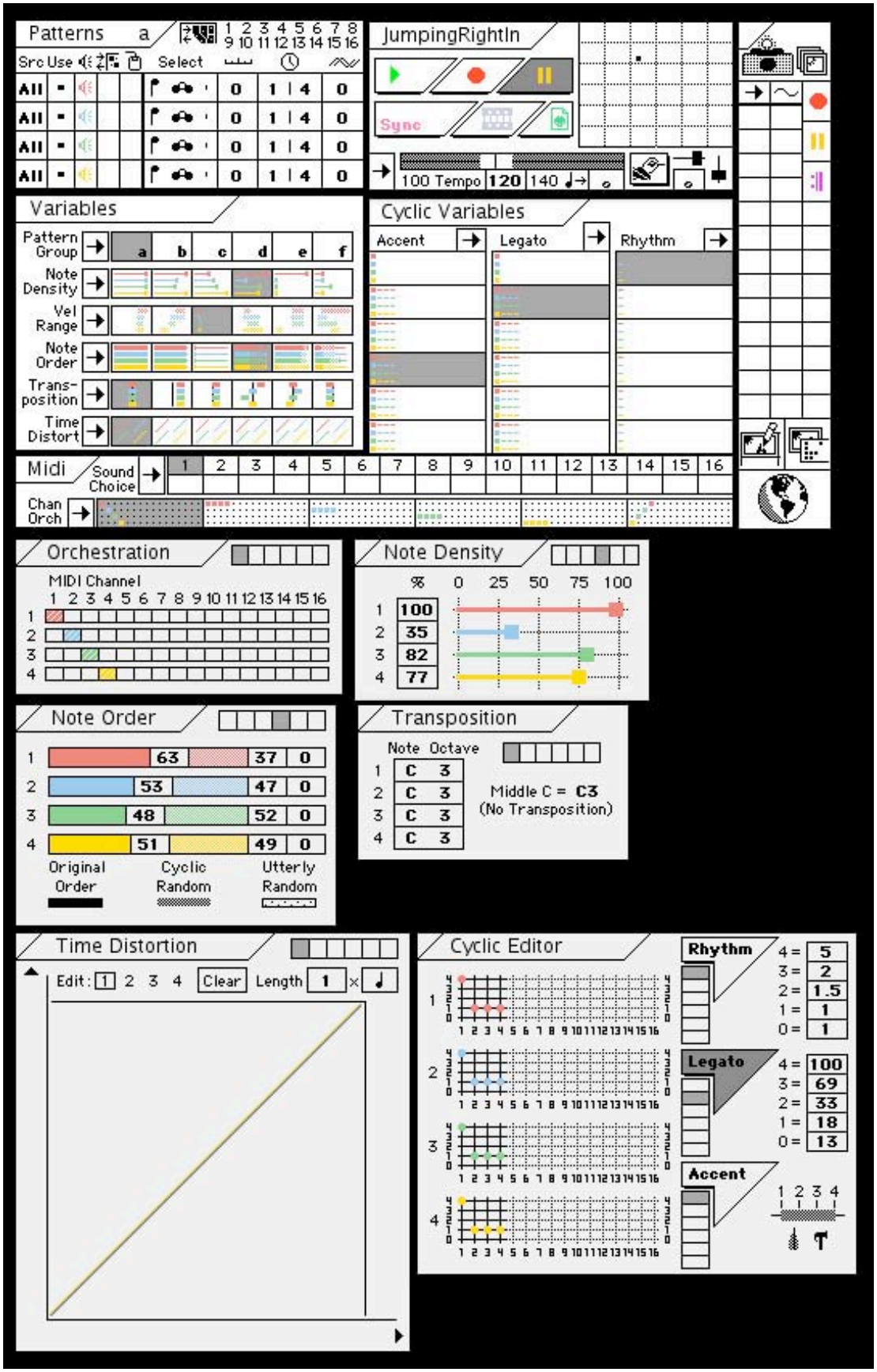


FIG. 11. Selected phrases from the output of the music composing machine. Set-up as of June 30, 1951, 4/4 time. Trinote probability derived from 11 Stephen Foster songs. Note, out of a total of 44 measures from the machine the following were selected, namely, 1 to 9, 13 to 25, 29 to 33, and 40 to 44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

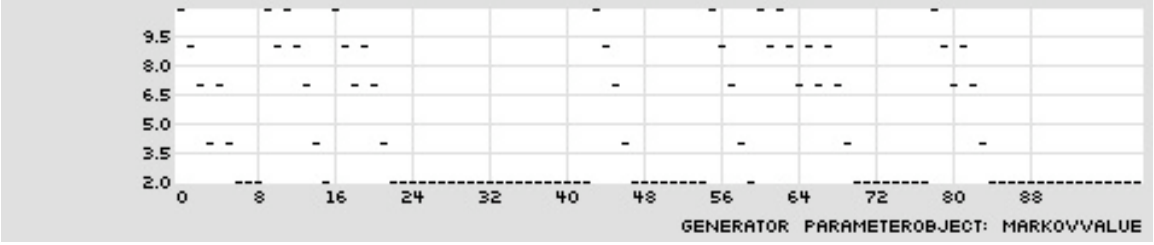
    tie r mp,a{8,1}b{4,3}c{4,7}d{4,13}a:{a=9|d=1}b:{a=5|c=1}c:{b=1}d:{c=1},(c,1)
  - eln; elh

10.15. Markov-Based Value Generation

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

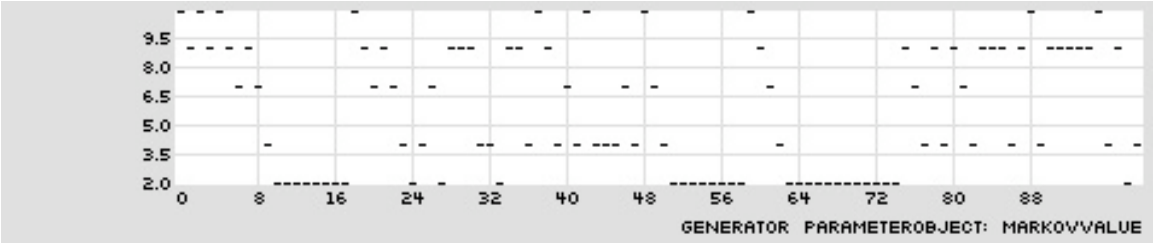
```
:: tpv mv
Generator ParameterObject
{name,documentation}
MarkovValue         markovValue, transitionString, parameterObject
                    Description: Produces values by means of a Markov transition
                    string specification and a dynamic transition order
                    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

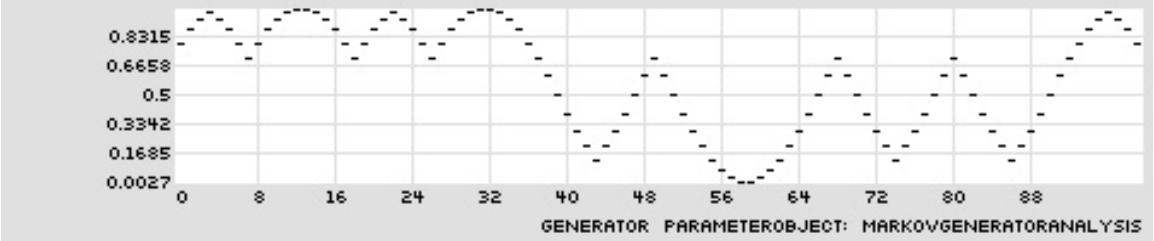
10.16. Markov-Based Combined Analysis and Generation

- 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

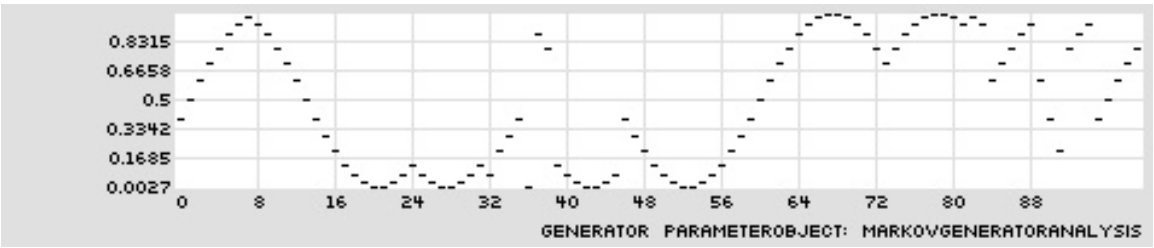
```
:: tpmap 100 mga,(ws,e,30),30,2,(c,1)
markovGeneratorAnalysis, (waveSine, event, (constant, 30), 0, (constant, 0),
(constant, 1)), 30, 2, (constant, 1)
TPmap display complete.
```



- 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

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