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

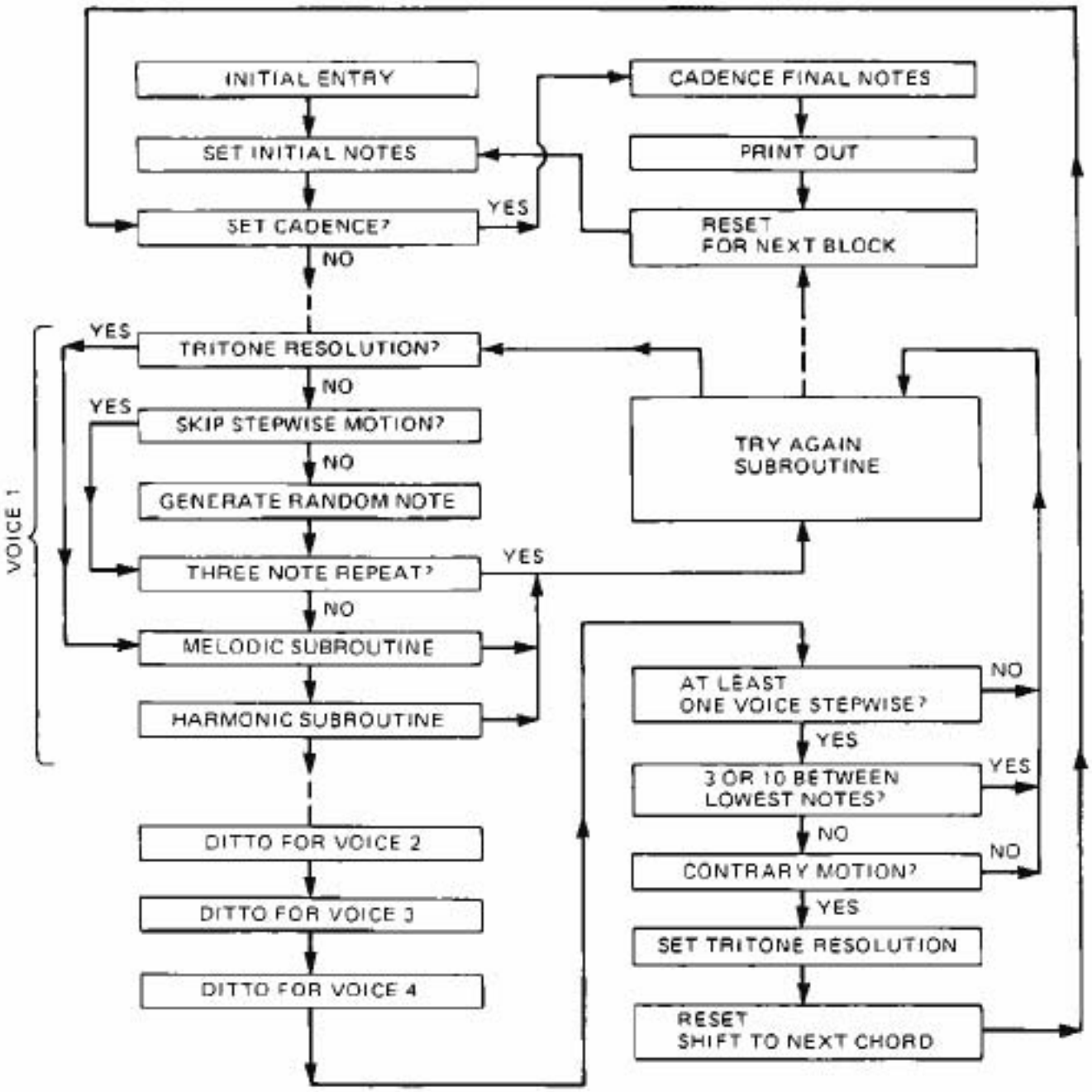


Figure 1.1
Experiment 2: Main routine for strict counterpoint.

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)

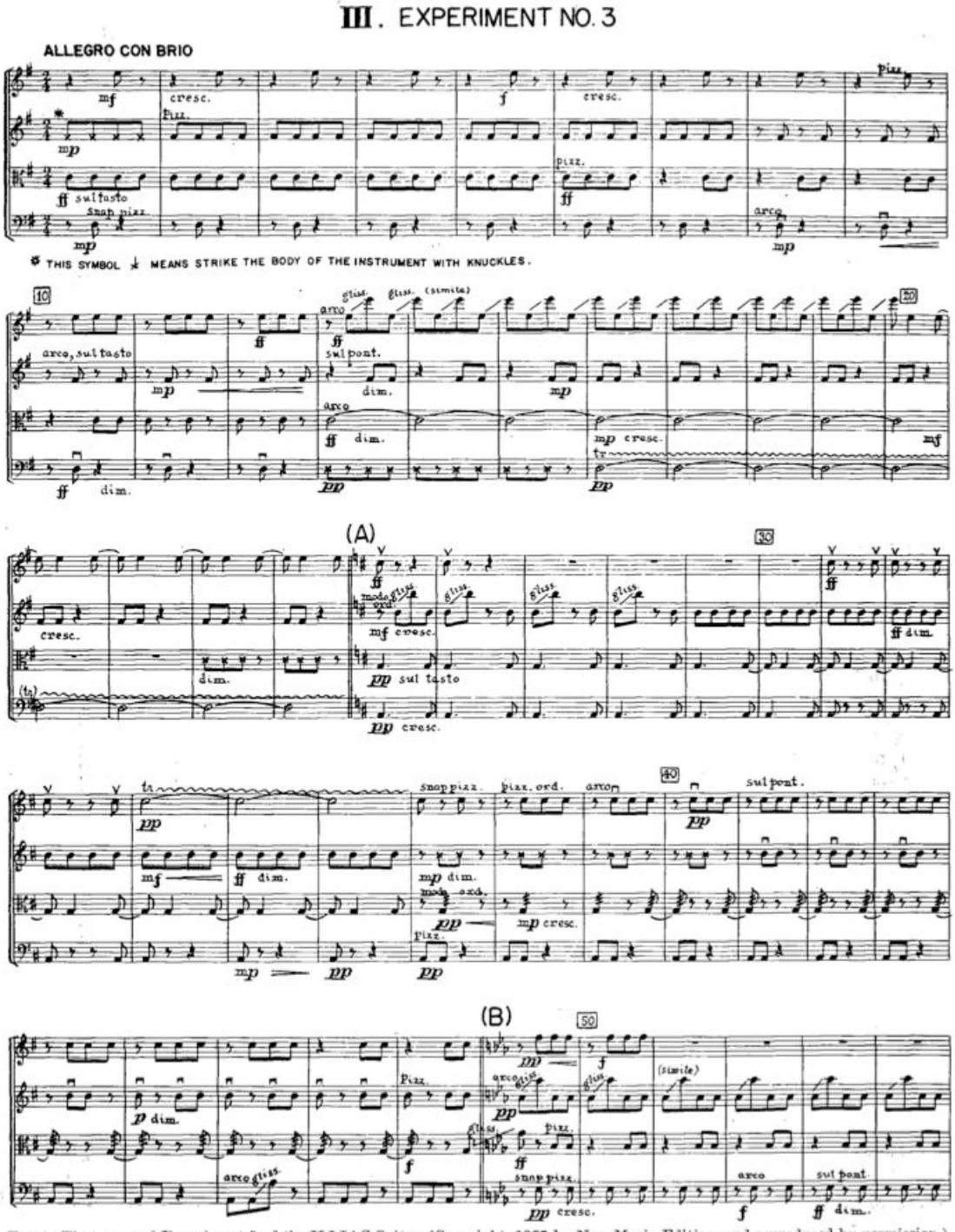


FIG. 3. First page of Experiment 3 of the ILLIAC Suite. (Copyright, 1957 by New Music Edition, and reproduced by permission.)

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

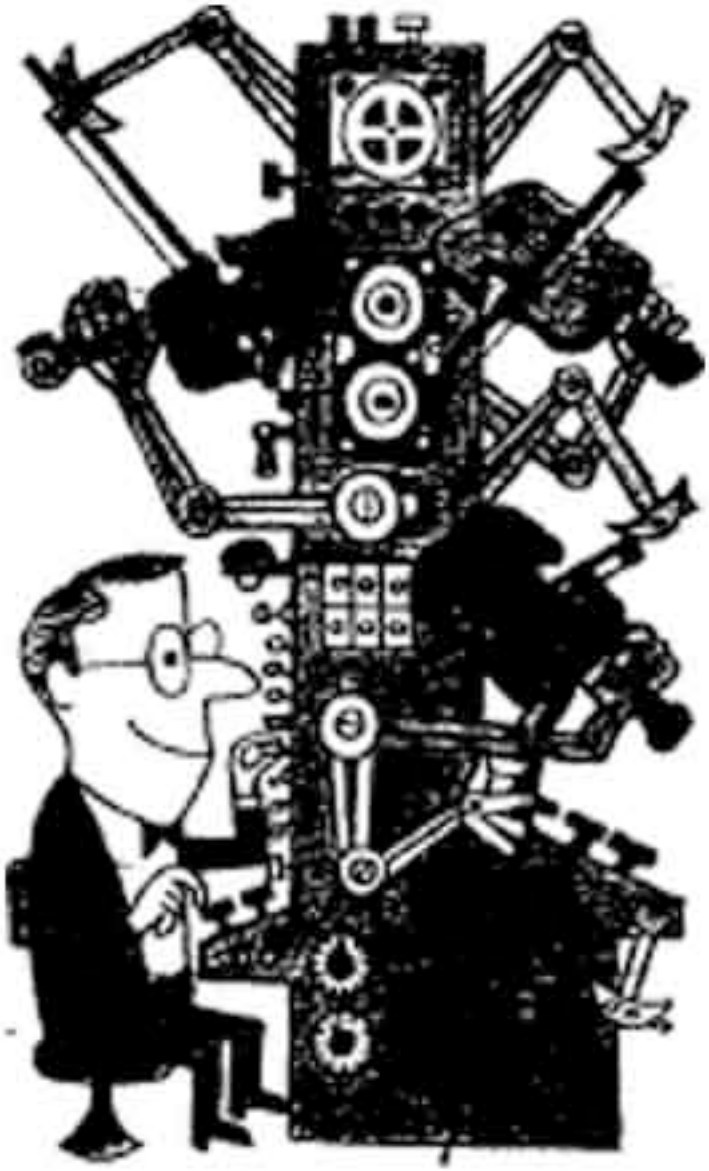
By Ed Cony
Staff Reporter of The Wall Street Journal

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"

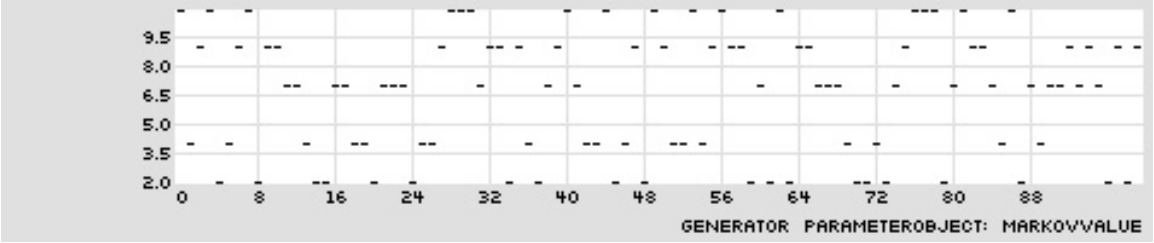
9.12. Zero Order Markov Chains as ParameterObjects

- 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}
```

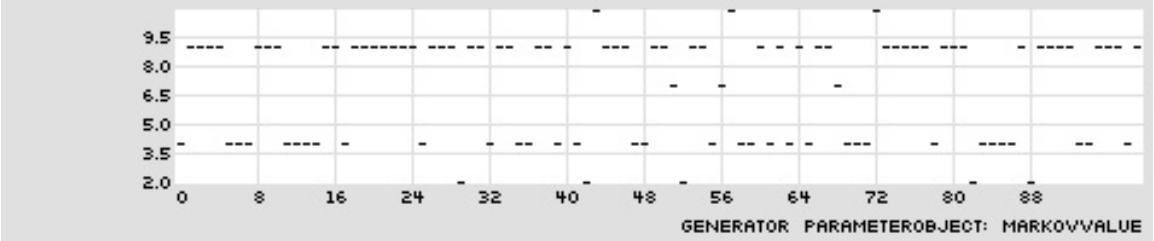
- The transition string
  - Two parts: symbol definitions and weights
  - Symbol definition: a{3}b{345}c{23.54}
  - Zero order weights: :{a=3|b=1|c=34}
- MarkovValue: zero order with equal weighting

```
:: 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

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