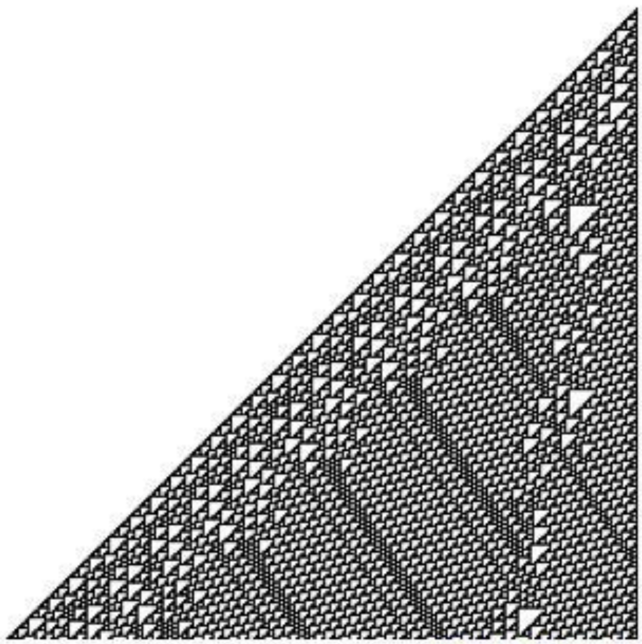
Chapter 17. Meeting 17, Approaches: Cellular Automata

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

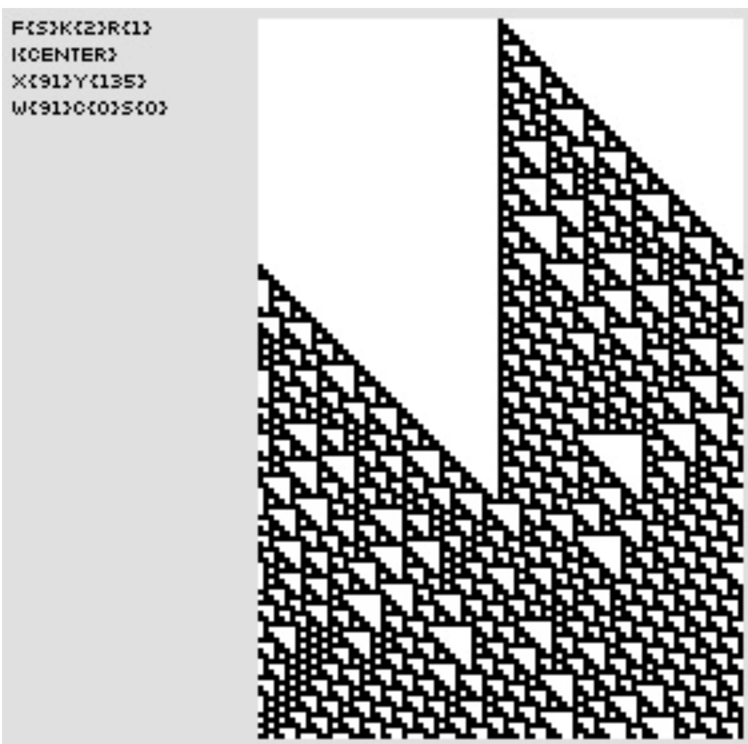
- String-based notation of CA forms
- Key-value pairs: key{value}

17.6. CA Types

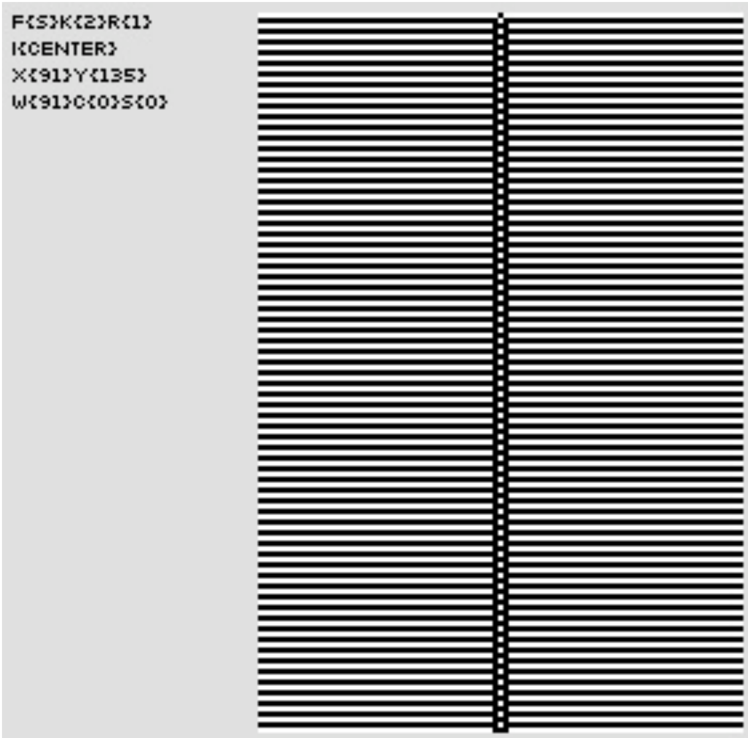
- Standard: f{s}

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}
```



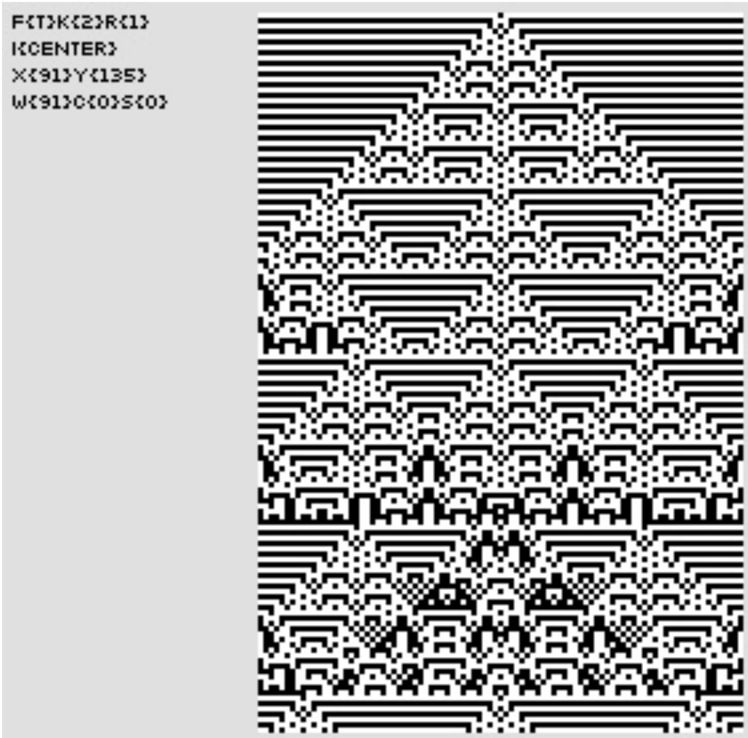

```
:: auca f{s} 379 0
f{s}k{2}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



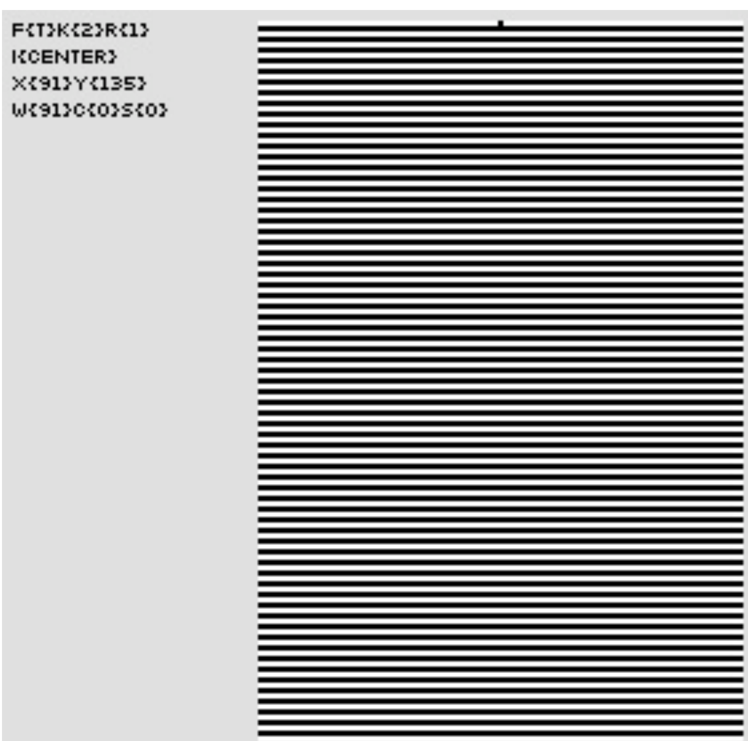
• Totalistic: f{t}

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}
```



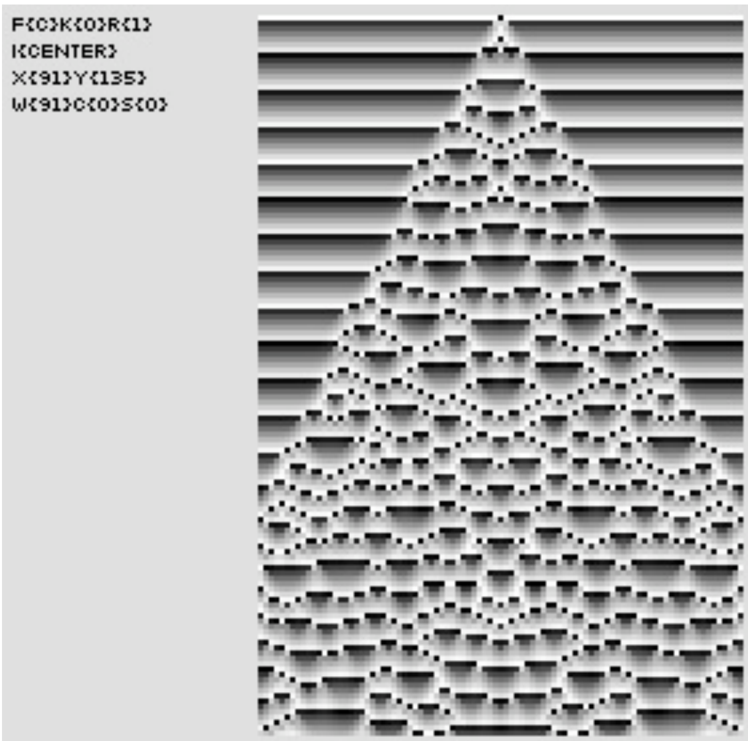

```
:: auca f{t} 39 0
f{t}k{2}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



• Continuous: f{c}

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

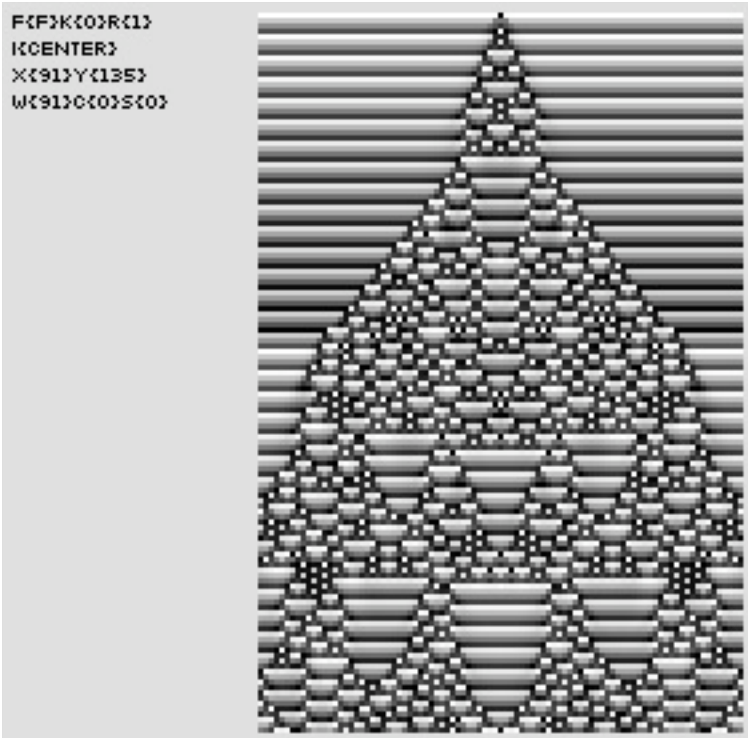
```
:: auca f{c} .8523 0
f{c}k{0}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



• Float: f{f}

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

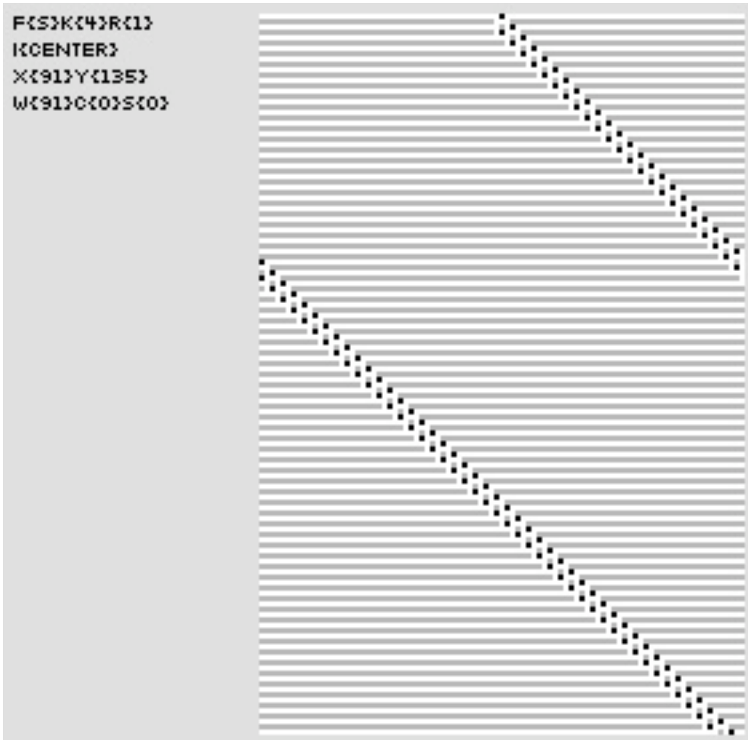
```
:: auca f{f} .254 0
f{f}k{0}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



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

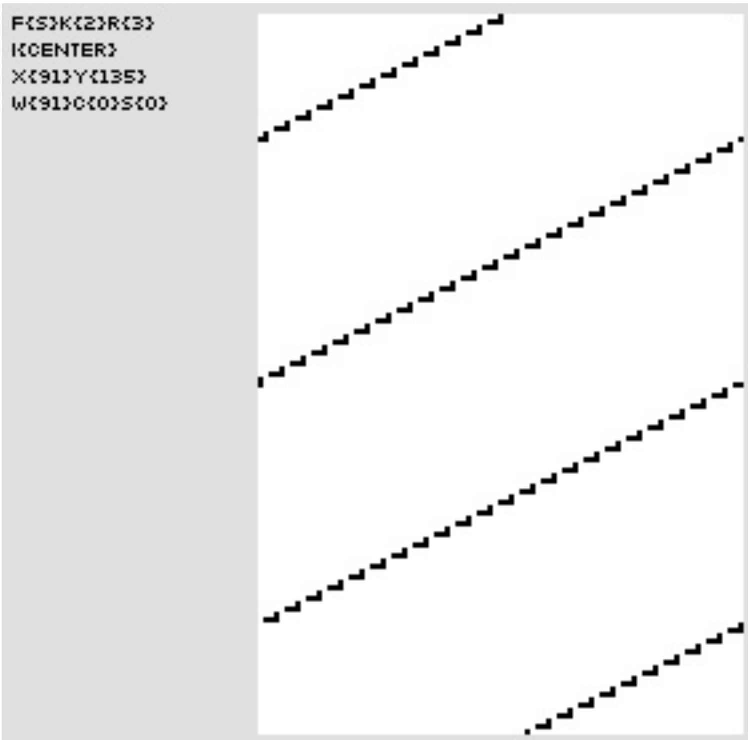
```
:: auca f{s}k{4} 3841 0
f{s}k{4}r{1}i{center}x{91}y{135}w{91}c{0}s{0}
```



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

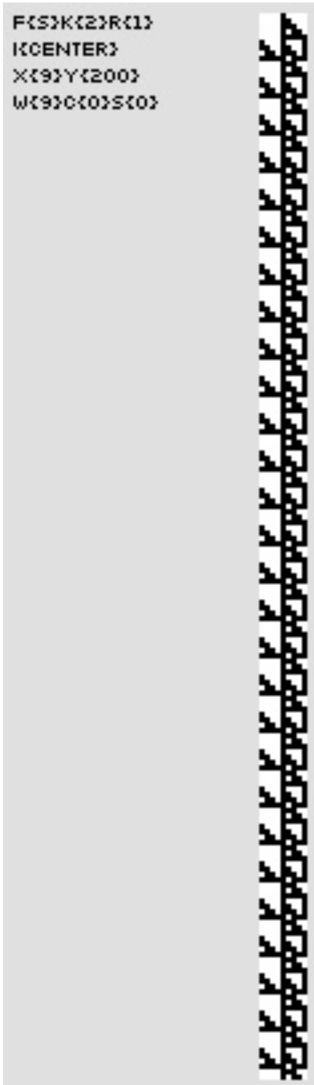
```
:: auca f{s}r{3} 380 0
f{s}k{2}r{3}i{center}x{91}y{135}w{91}c{0}s{0}
```



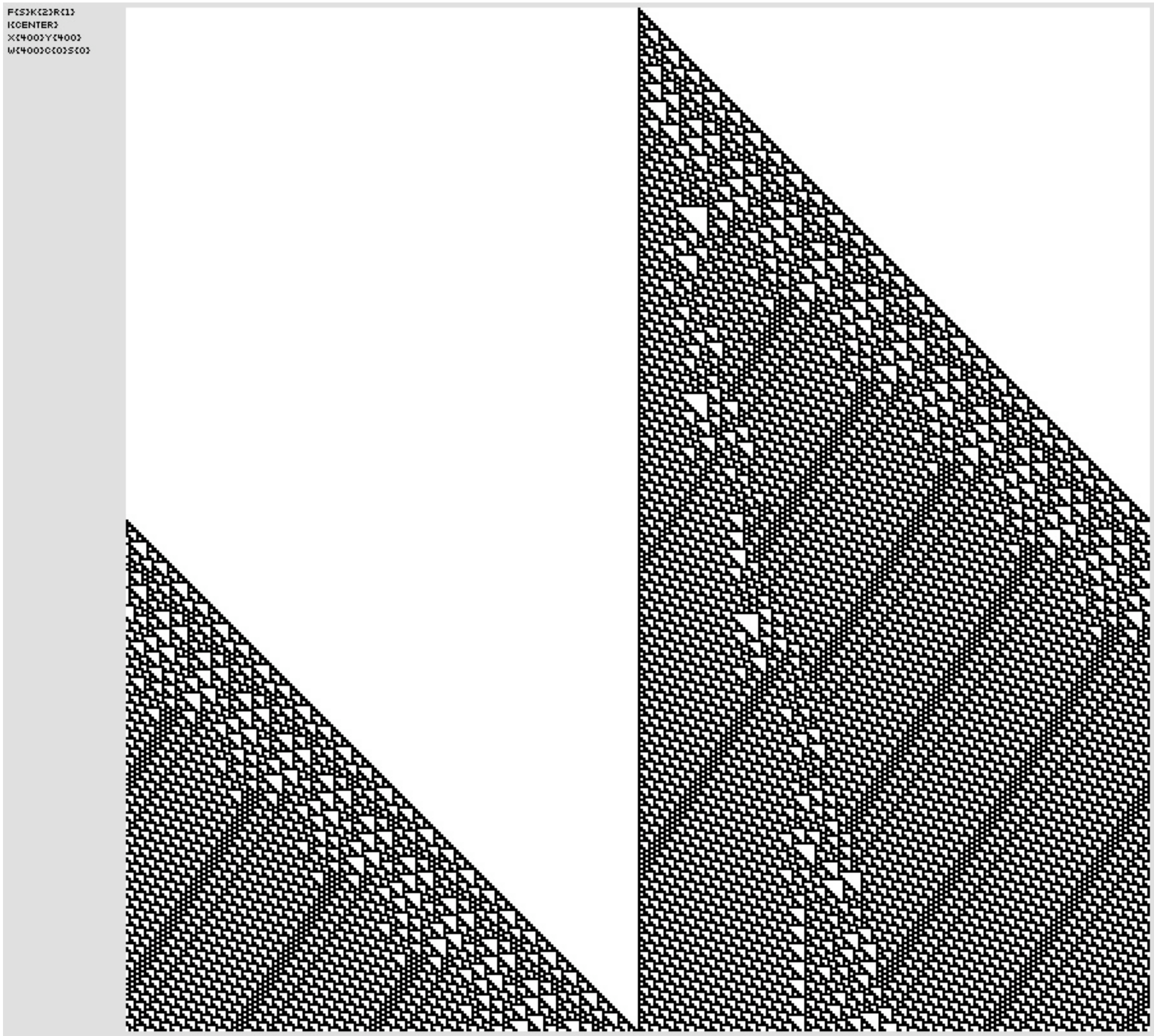
17.9. Size, Orientation, and Presentation

- 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}
```




```
:: 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

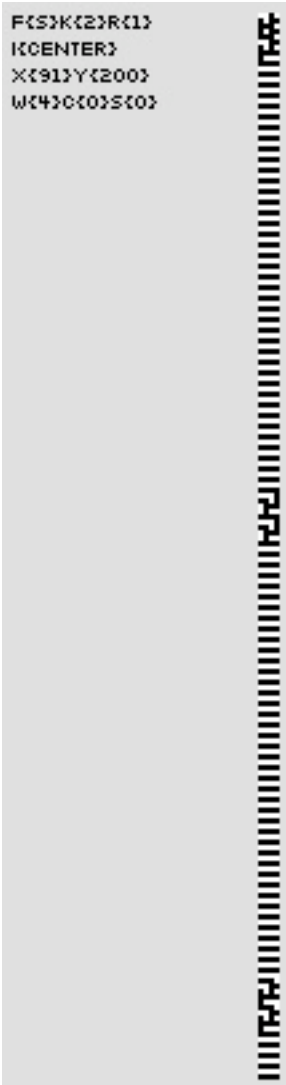
  Width, w{}, is the number of exposed cells

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

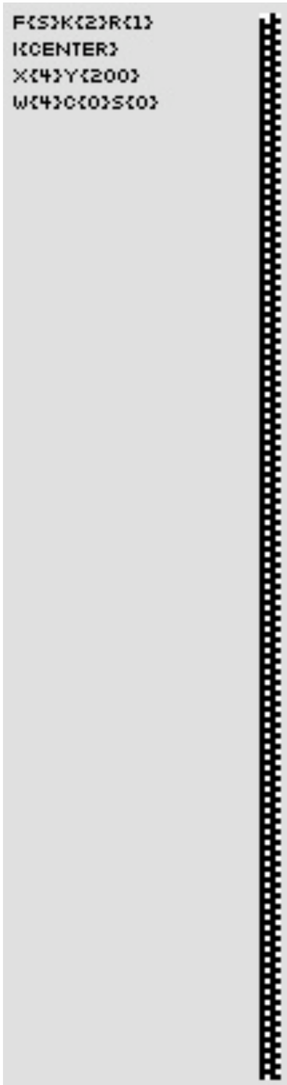
  Skip, s{}, is the number of rows neither displayed nor counter in *y*.

- 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}
```



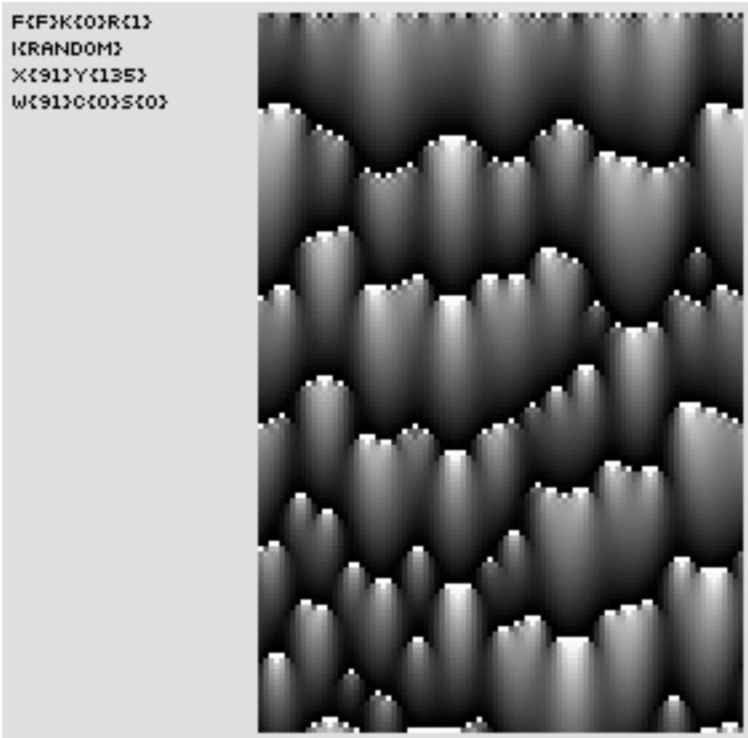

```
:: auca f{s}x{4}y{200} 381 0
f{s}k{2}r{1}i{center}x{4}y{200}w{4}c{0}s{0}
```



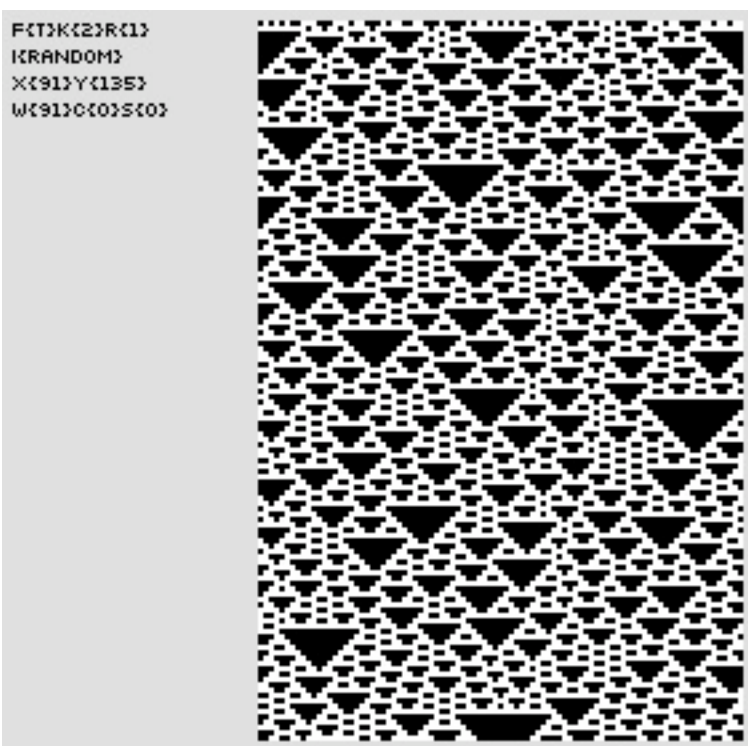
17.10. The Initial Row

- 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}
```

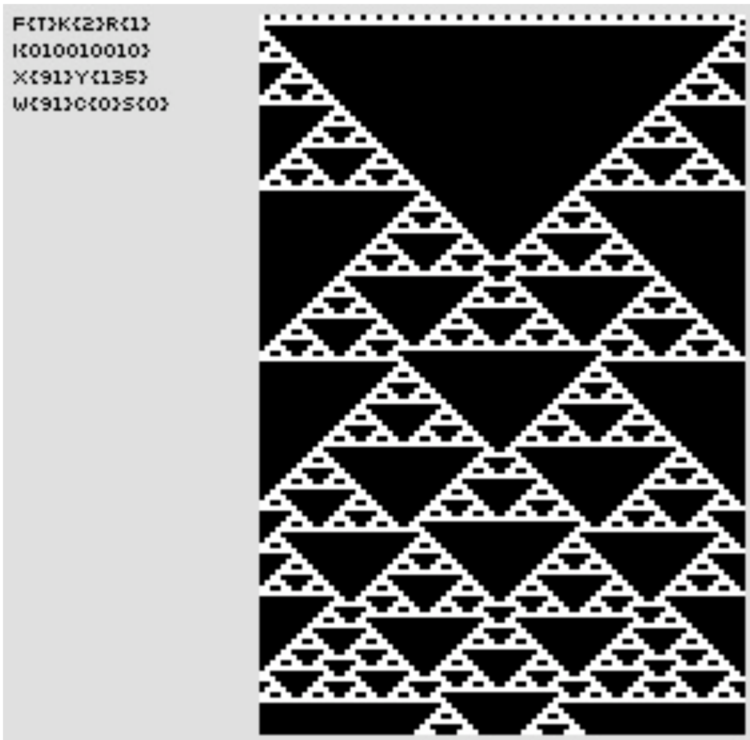


:: auca f{t}i{r} 201 0
f{t}k{2}r{1}i{random}x{91}y{135}w{91}c{0}s{0}



- 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

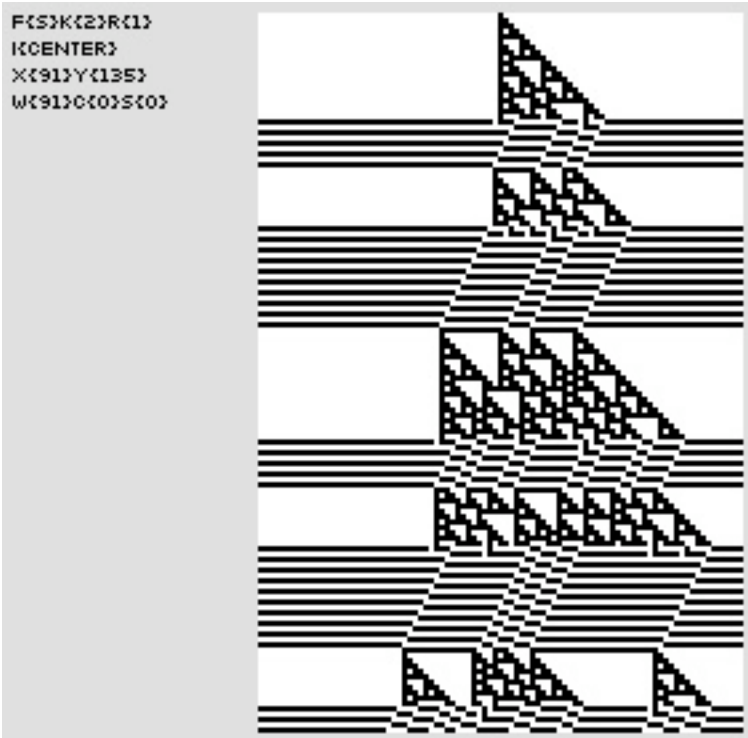
- Rule: a floating or integer value

  Wolfram offers standard encoding of rules as integers

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

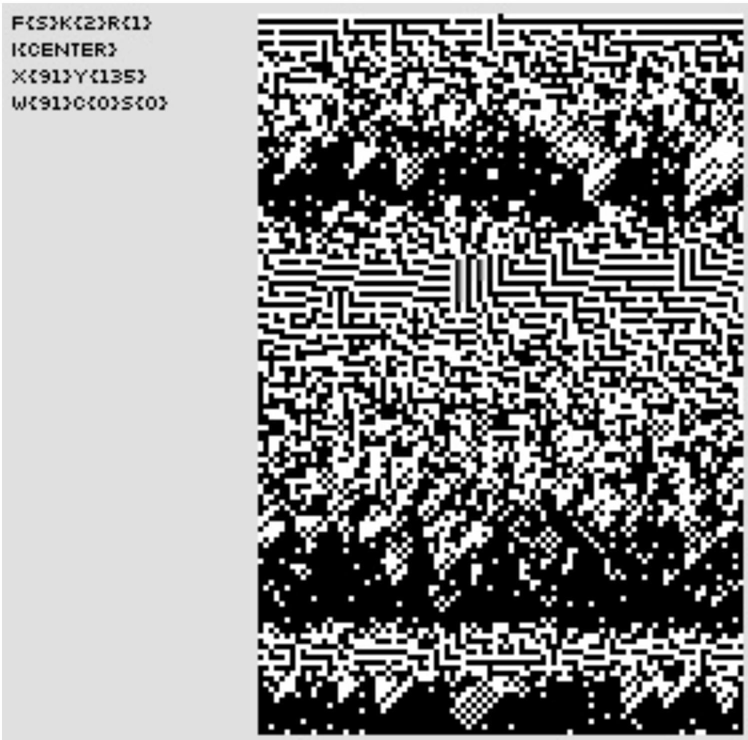
- PO applied to the rule value of CA

```
:: 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.

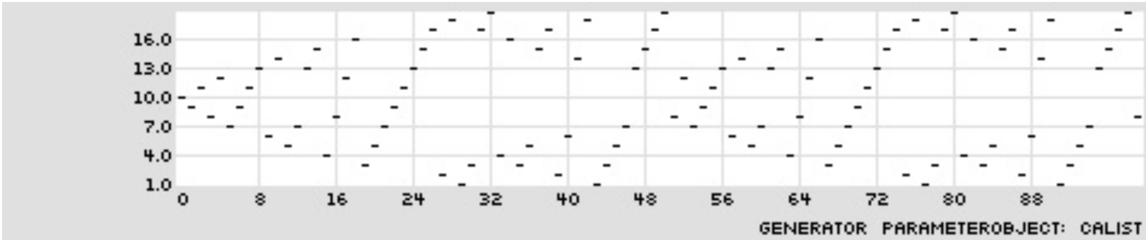
17.15. The CA as ParameterObject

- 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*

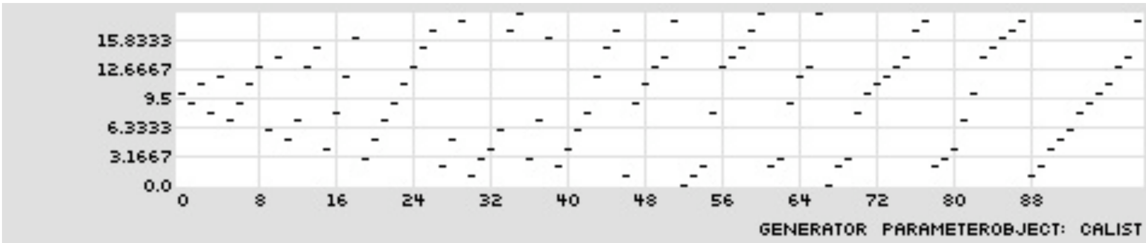
    tie d0 c,0
  - *select pitches from Multiset using CaList*

    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*

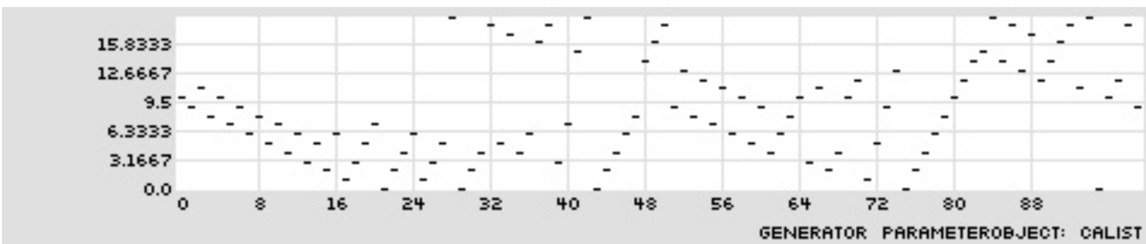
    tie d3 c,1
  - eln; elh
- CaList with rule 90 and flatRowIndexActive; a smaller x is used to reduce index values; adding mutation

```
:: 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

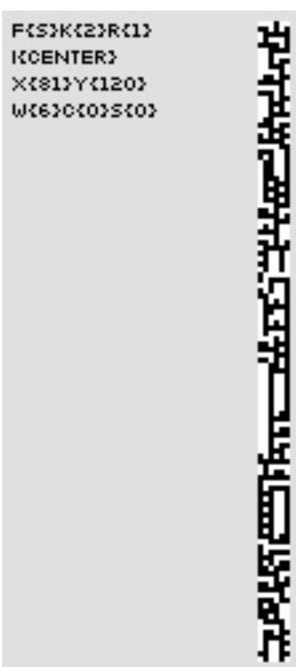
- Narrow regions of bent CA offer interesting variation of few values

- 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}
```

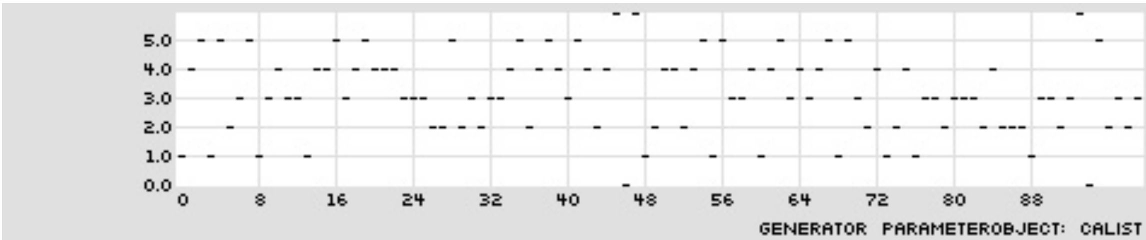
- A a narrow width of a CA with a small constant mutation

```
:: 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}
```



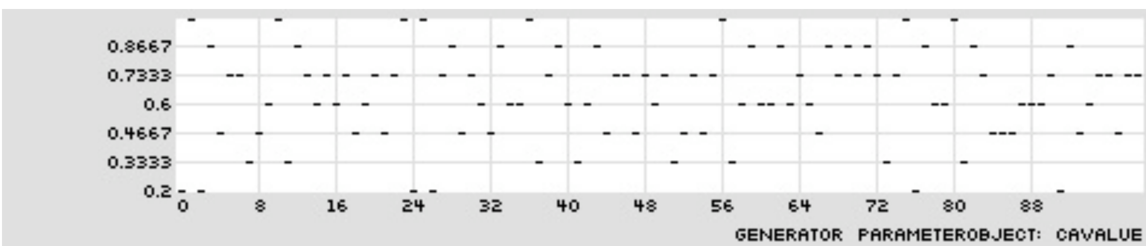
- Using CaTable and sumRowActive, we can get a dynamic collection of small integer values

```
:: 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.
```



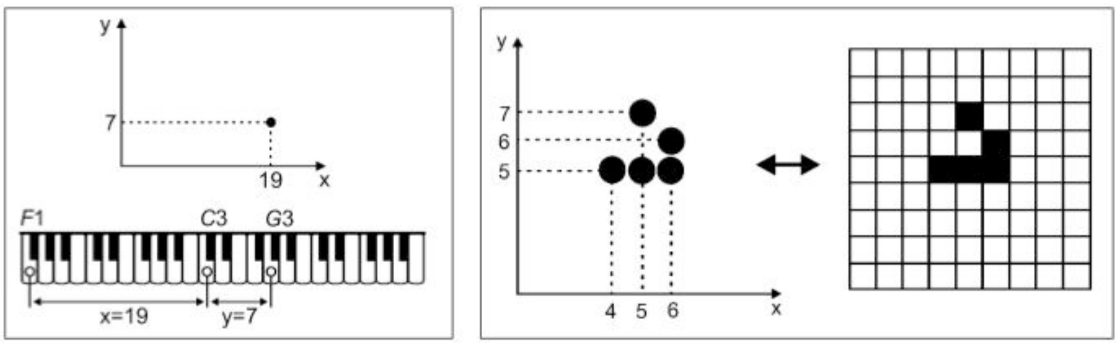
- Command sequence using TM Harmonic Assembly:
  - emo mp
  - tin a 47
  - *set the multiplier to the integer output of CaList*

    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?



- 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?