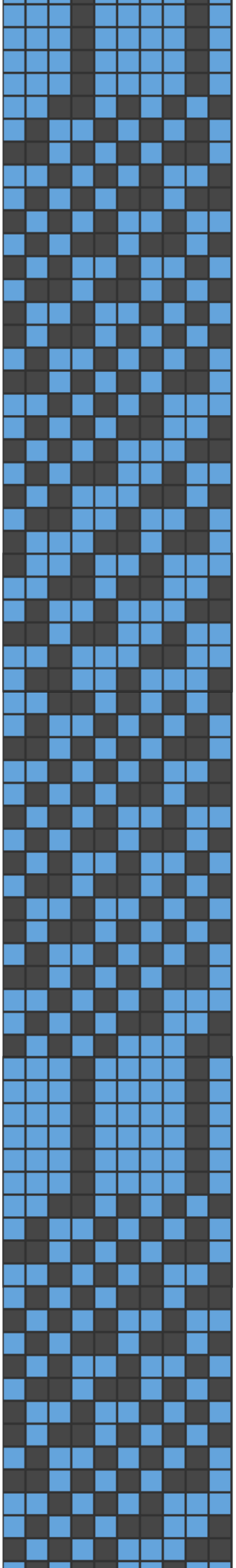
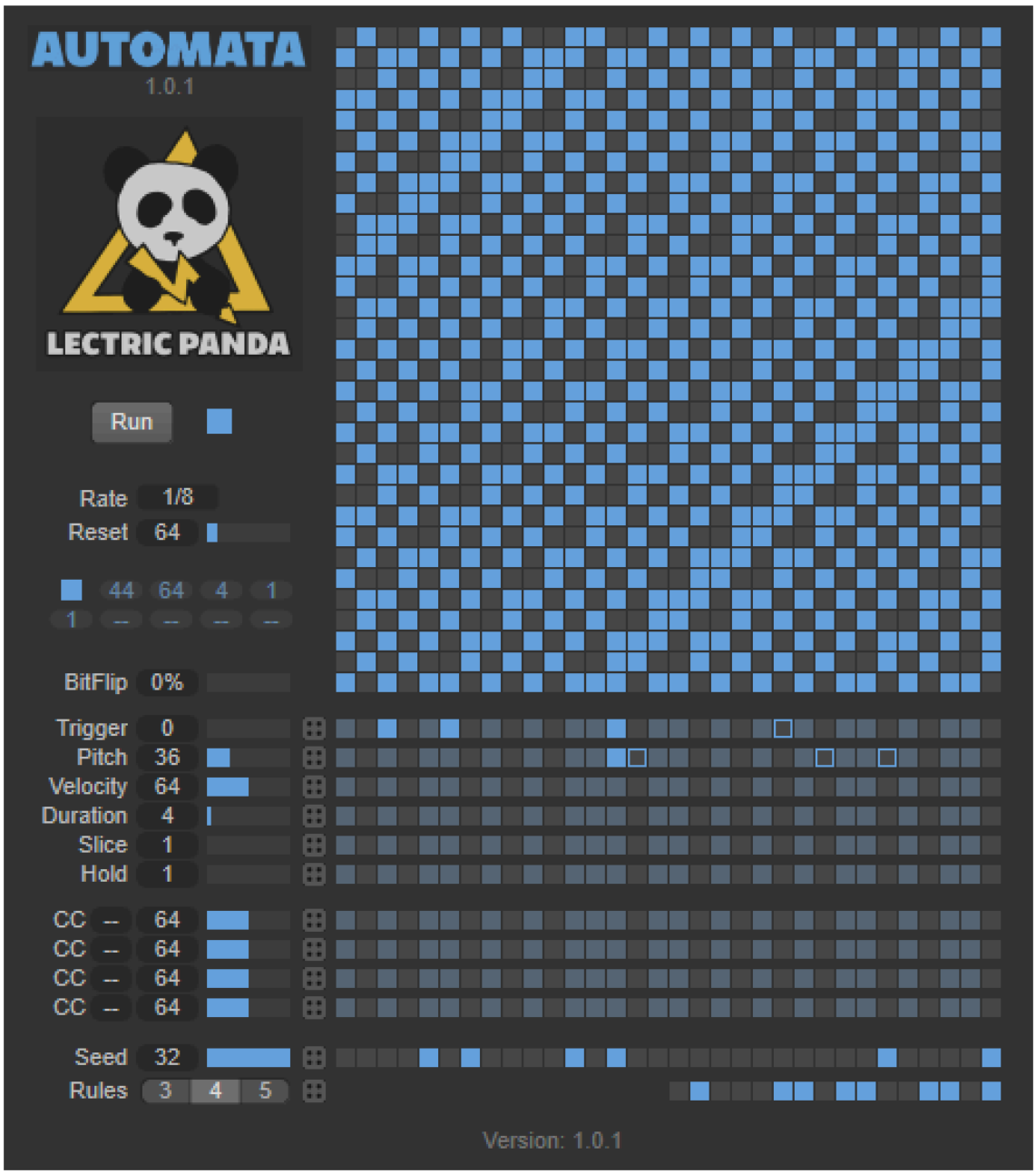
AUTOMATA

CELLULAR AUTOMATA SEQUENCER

OPERATION MANUAL





INTRODUCTION

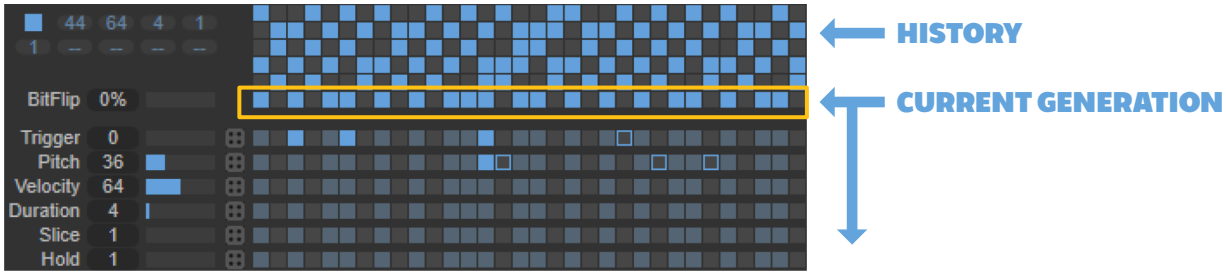
AUTOMATA is a VST3 Plugin MIDI Sequencer / Generator. It uses each evolutionary step of a Cellular Automata to generate parameters of the generated MIDI notes. The cyclic nature of CA evolution can create musically interesting patterns.

CELLULAR AUTOMATON

A cellular automaton consists of a regular grid of cells, each in one of a finite number of states, such as on and off. The grid can be in any finite number of dimensions. For each cell, a set of cells called its neighborhood is defined relative to the specified cell. An initial state (time t = 0) is selected by assigning a state for each cell. A new generation is created (advancing t by 1), according to some fixed rule (generally, a mathematical function) that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighborhood.

Cellular automaton (2023, Nov 18) In *Wikipedia.* https://en.wikipedia.org/wiki/Cellular_automaton

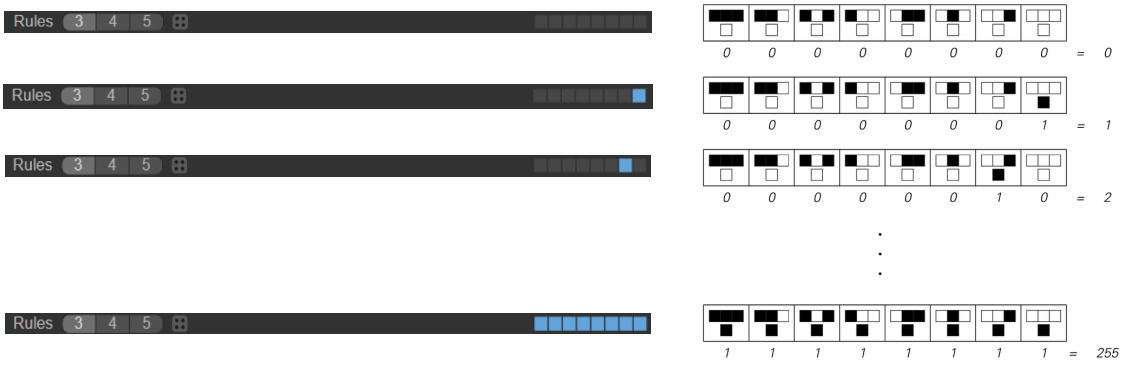
AUTOMATA uses one dimensional cellular automation. Each step creates a new row and shifts the history upwards. Only the current generation is used for MIDI parameters. The state of the current step are projected downward to the MIDI value modifiers. The history is only used for visualization purposes.



A new generation is produced on each step of the system. The timing of the step depends on the **Rate** parameter. On the step, a cell's neighbors are evaluated based on the **Rules**, the rules will determine the new state.

The active bits in the **Rules** correspond to the new states for a given rule. Pictured are examples for rules with a neighborhood of size K=3.

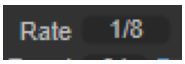
Image: https://www.wolframscience.com/nks/p53--more-cellular-automata/

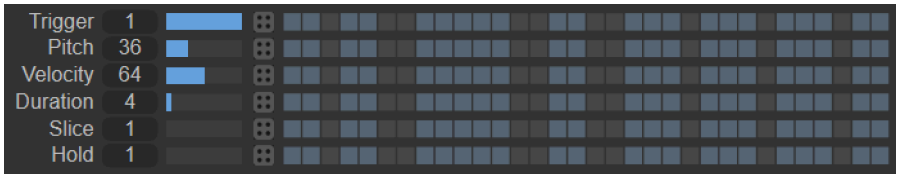


OPERATION

You don't need to fully understand the details of Cellular Automata to use **AUTOMATA**. While the subject matter can be very deep and interesting, what we care about here is generating some interesting MIDI data. For this, we can get a very long way with the basics of operation and the randomize buttons.

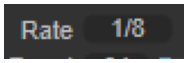
At the very core of **AUTOMATA** is a MIDI note / pulse generator. The parameter **Rate** sets the frequency that ticks the system.

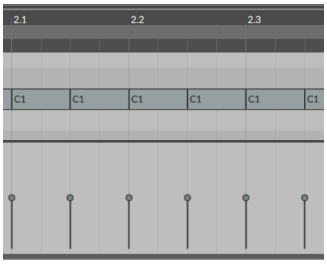




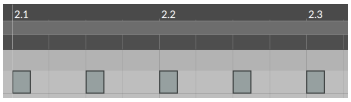
If we look at the above image, there are ***_no_*** bit modifiers active and **Trigger** is **1.** We can expect a midi note to be generated every **1/8th** note, with a **Pitch** value of 36, **Velocity** of 64.

Duration is specified in fractions (1/4ths) of the base **Rate.** For a base **Rate** of 1/8th, duration of 1 would generate a 1/32nd note every 1/8th. A duration of 2 would generate a 1/16th every 1/8th. A duration of 4 is the full 1/8th note.

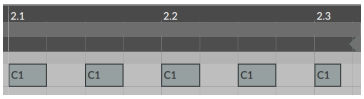




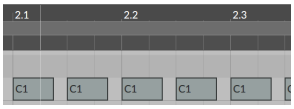
BIP BIP BIP BIP 8th NOTE RATE
DURATION: 4



DURATION: 1

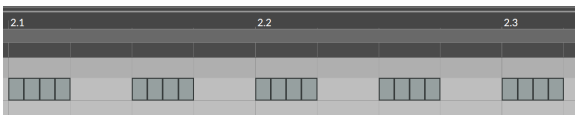


DURATION: 2



DURATION: 3

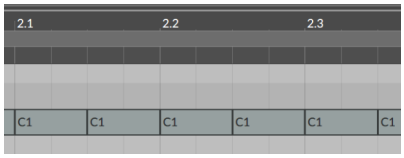
Slice will slice up the note into equal parts over the note's **Duration**.



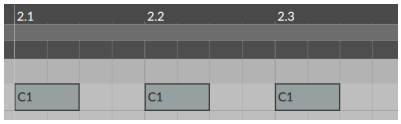
DURATION: 2
SLICE: 4

DURATION: 3
SLICE: 2

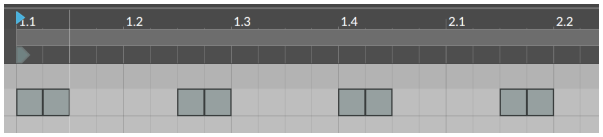
Hold can delay the next step / generated note. A **Hold** of 1 is the normal, no-delay base **Rate**. A Hold of 2 will add an extra period, nothing will be generated, and the CA won't step.



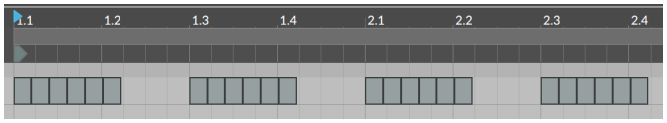
DURATION: 4
HOLD: 1
SLICE: 1



DURATION: 4
HOLD: 2

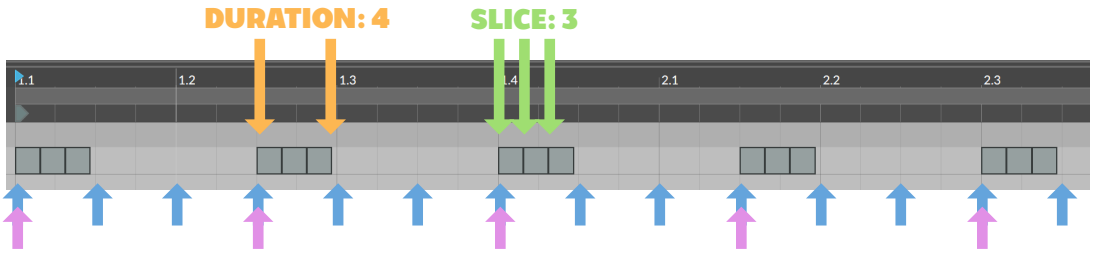


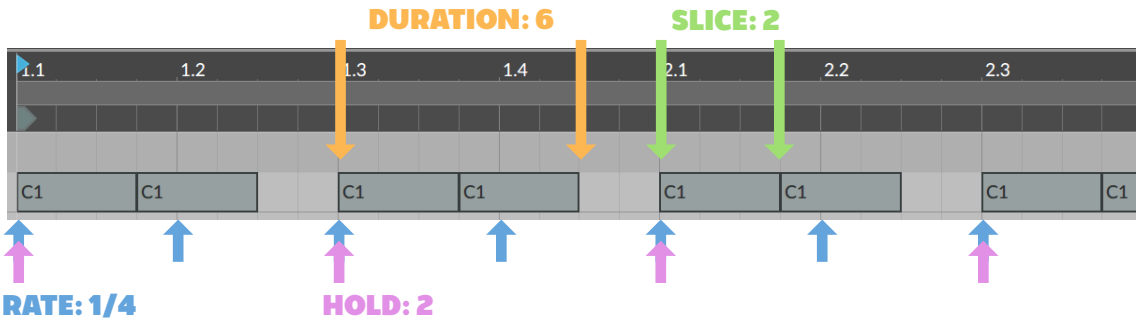
DURATION: 4
HOLD: 3
SLICE: 2



DURATION: 10
HOLD: 4
SLICE: 6

To summarize the three parameters that effect timing: Base **Rate** determines how often the event fires. **Duration** is how long the event lasts. **Slice** determines how many individual notes are generated. **Hold** controls how many events are skipped.

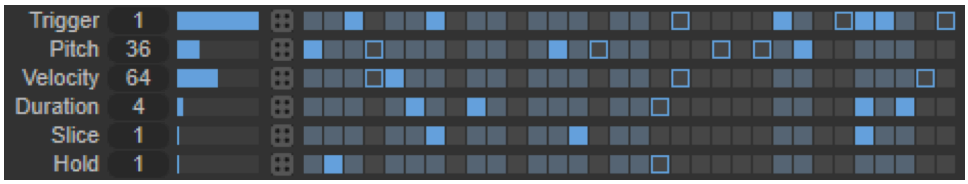




MODIFIER VALUES

Values are evaluated on each step. The base values on the left are added to any active bits selected on the right.

BASE VALUES + BIT MODIFIERS = FINAL STEP VALUE



Clicking on a bit will select it for evaluation. If the cellular automata's bits are active, they will be added to the base value.

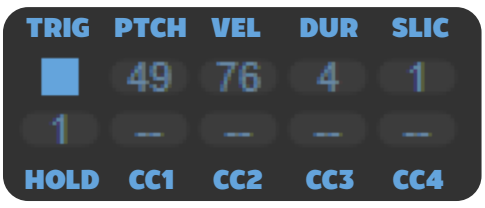
PITCH 36 + 000000 b = 36
VELOCITY 64 + 00000b = 64

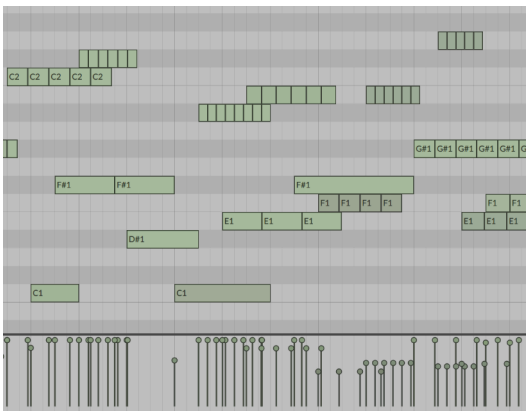
PITCH 36 + 000000 b = 36
VELOCITY 64 + 00000b = 64

PITCH 36 + 000001 b = 36 + 1 = 37
VELOCITY 64 + 00110b = 64 + 6 = 70

PITCH 36 + 001101 b = 36 + 13 = 49
VELOCITY 64 + 01100b = 64 + 12 = 76

The final calculated values are displayed to the left of the automata history. With modifiers, each step can produce different pitch, velocity, duration, slices, and hold values.

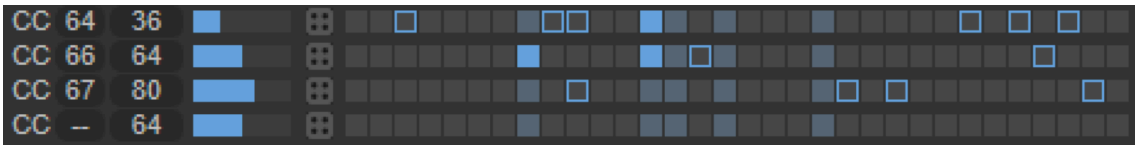




Binary	Dec
00000	0
00001	1
00010	2
00011	3
00100	4
00101	5
00110	6
00111	7
01000	8
01001	9
01010	10
01011	11
01100	12
01101	13
01110	14
01111	15
10000	16
10001	17
10010	18
10011	19
10100	20
10101	21
10110	22
10111	23
11000	24
11001	25
11010	26
11011	27
11100	28
11101	29
11110	30
11111	31

MIDI CC

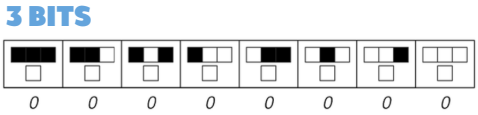
MIDI CCs can be generated on each trigger if enabled. The values are calculated with the same base + bit modifiers. The CC number can be set. CC number 128 will disable that row.



MORE AUTOMATA

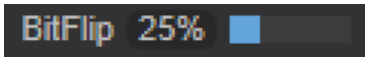
The entire Automata system can **Reset** every number of steps. A **Reset** value of 0 will disable resetting the system. On each reset, the **Seed** value is placed as the current CA state.

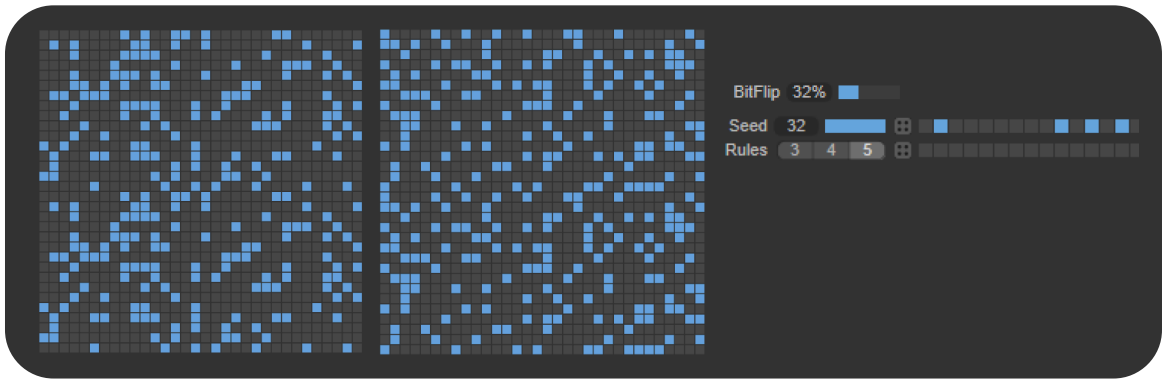
The **Rules** 3,4,5 sets the number of bits / number of neighbors for each rule. The **Rule** bits set the result state from 11111b (left most) to 00000b (right most).



BITFLIP

The **Bitflip** percentage determines the chance that any bit will flip on the new step. The randomizer is re-seeded from the **Seed** value on every **Reset** so the randomizing is cyclic. For pure random generation, set a **Seed** and the **Rules** empty.





THANK YOU

Thank you for trying **AUTOMATA!**