SZCZ's AUDIO ADVENTURES

CRACKLEFIELD

CELLULAR GAME SEQUENCER

USER'S GUIDE

www.fairlyconfusing.net

Licence and technical information

Cracklefield is a virtual instrument for Native Instruments Kontakt sampler, you need full version of Kontakt 5.8.1 or newer to run it.

Instrument interface is fairly large, it takes 1000x770 pixels inside Kontakt. If you want to be able to access the whole interface without scrolling, be sure to use screen resolution of at least 1280x1024.

Cracklefield

Cracklefield is experimental musical device, a fusion between a sequencer and cellular automaton. While regular sequencers operate on series of data lines, Cracklefield is using data grid, called the field. It can animate a number of cursors travelling the field. Cursors can move horizontally, forward or backward, vertically, up or down, or diagonally, interacting with the field data, field boundaries, obstacles and, what's most interesting, with each other.

The instrument generates more or less complicated, evolving note patterns by running a kind of mathematical simulation.

It comes with a set of unusual, mostly acoustic sounds, which can be easily expanded by user samples or copying zones from another instrument.

Contents:

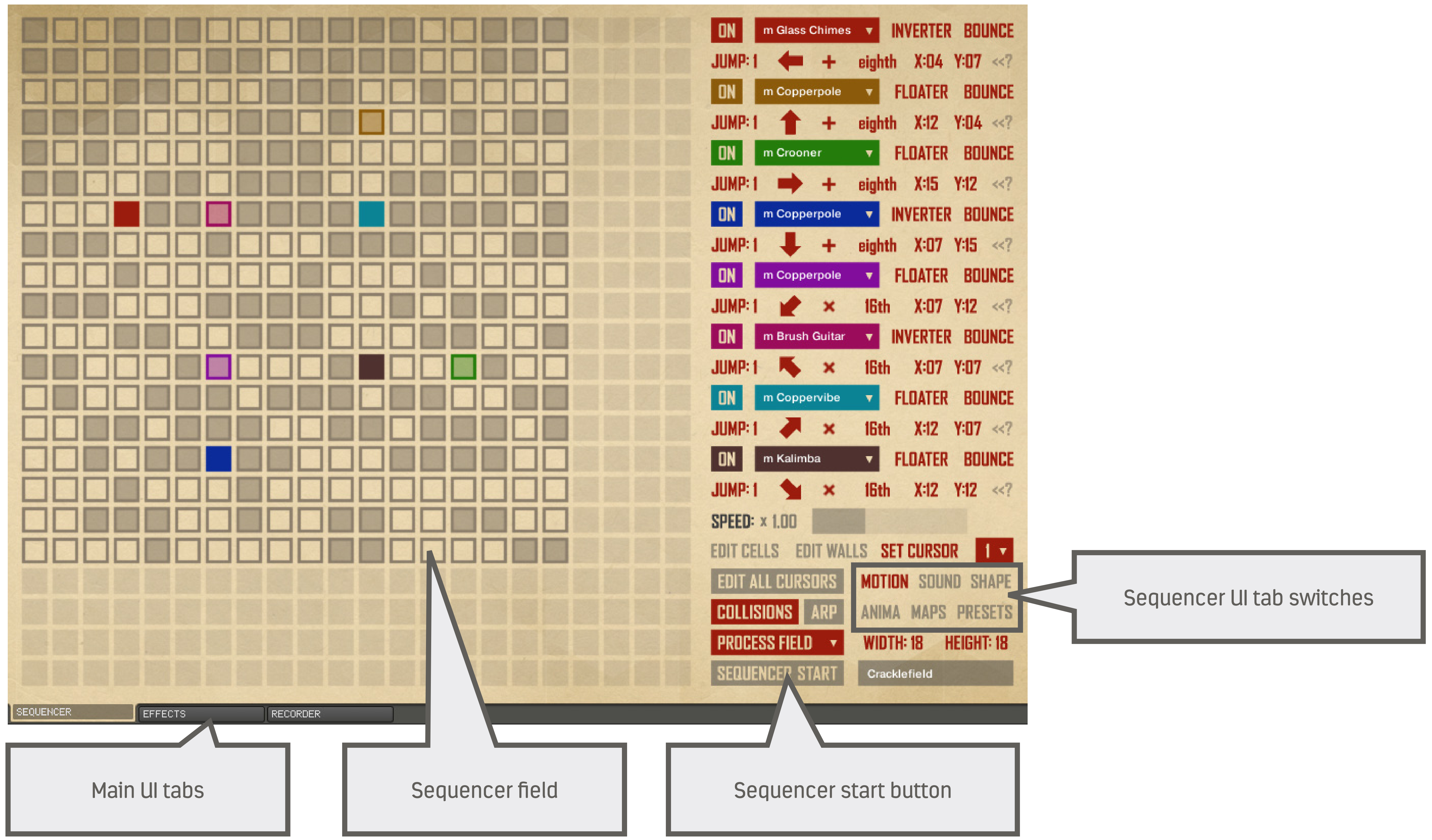
User interface basics

There are three main interface tabs: SEQUENCER, EFFECTS and RECORDER.

SEQUENCER tab contains instrument main engine controllers. EFFECTS tab (obviously) provides controllers for included audio effects. In the last tab, RECORDER, you can find MIDI note recorder with drag'n'drop functionality, as well as various controller options.

Sequencer part of the interface has several sub-tabs: MOTION, SOUND, SHAPE, ANIMA (animator), MAPS and PRESETS.

SEQUENCER START button starts/stops the sequencer. MIDI keys can be used to play along with the generated sequence. There's also "arpeggiator mode", where the sequencer is running for as long as there are keys being held on MIDI keyboard, using chord notes to determine which notes to play in the sequence. (Note that when ARP is on, SEQUENCER START button has no function, sequencer starts/stops automatically).



Programming sequencer field

Sequencer field is rectangle cellular grid. A cursor generates a sound when it travels onto a filled cell. The field can be programmed manually, or filled automatically.

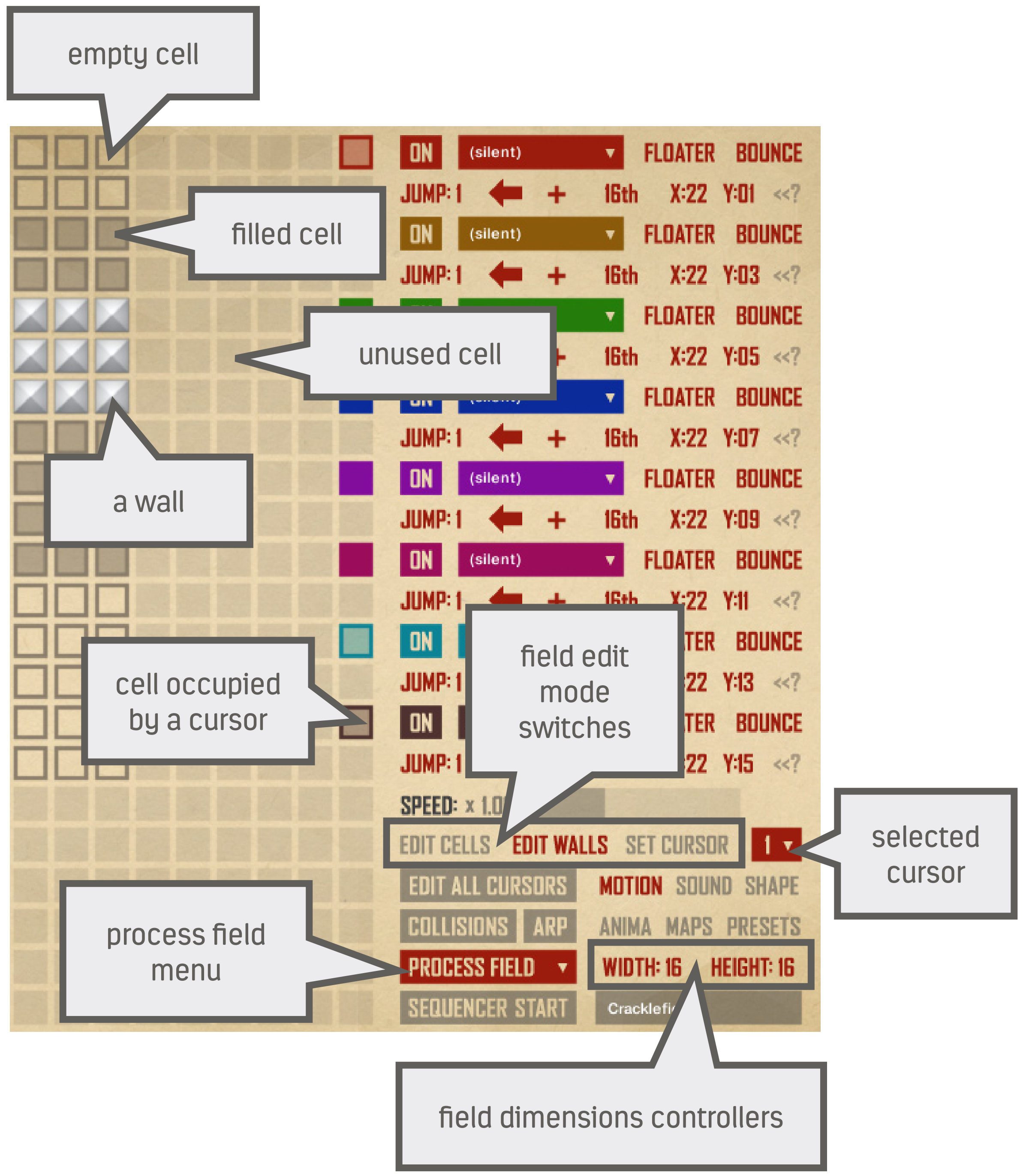
Maximum field size is 22x22 cells. It can be trimmed down to 4x4 cells. To set field dimensions, use WIDTH and HEIGHT controllers (which act as vertical sliders). Unused portion of the field is greyed out (hidden) and it is not used in simulation.

There are three field edit mode switches: EDIT CELLS, EDIT WALLS and SET CURSOR.

Set EDIT CELLS to program cell data, that is, switch the cells on/off (filled/empty). To do so, simply click on a cell. When programming cells, three key-switches can be used. Hold "shift", while clicking, to apply changes to entire field row. Hold "alt" to apply changes to entire column. Hold "control" with "alt" and/or "shift" to swap values (empty becomes filled and vice versa) in whole row/column.

Use EDIT WALLS switch, to place or remove walls on the field. A wall acts as an obstacle for cursors, which will (depending on cursor mode) bounce off them, or teleport to the other side.

SET CURSOR mode can be used to conveniently position cursors in the field. When this mode is active, clicking on a field cell will place selected cursor at that cell. There's SELECTED CURSOR controller, next to SET CURSOR switch. Click on it, to select cursor from a drop-down menu. Alternatively, you can select a cursor by clicking on the cell, it is set at, which is more convenient than using the menu.



Field contents can be generated/modified automatically; to see the list of available options, click on PROCESS FIELD drop-down menu.

<u>Options:</u>

Clear field – set all cells to empty.

Invert field – swap values for all cells (filled cell becomes empty and vice versa).

Clear walls – remove all wall cells.

Generate field pattern – generate a random pattern, each row will be filled with repeating sequence.

Randomise field – pick random cells and swap their values, can be applied to 10, 20 or 50% of cells.

Symmetrise (double mirror) – clones upper-left quarter of the field. First it is mirrored horizontally, then upper half of the field is mirrored vertically.

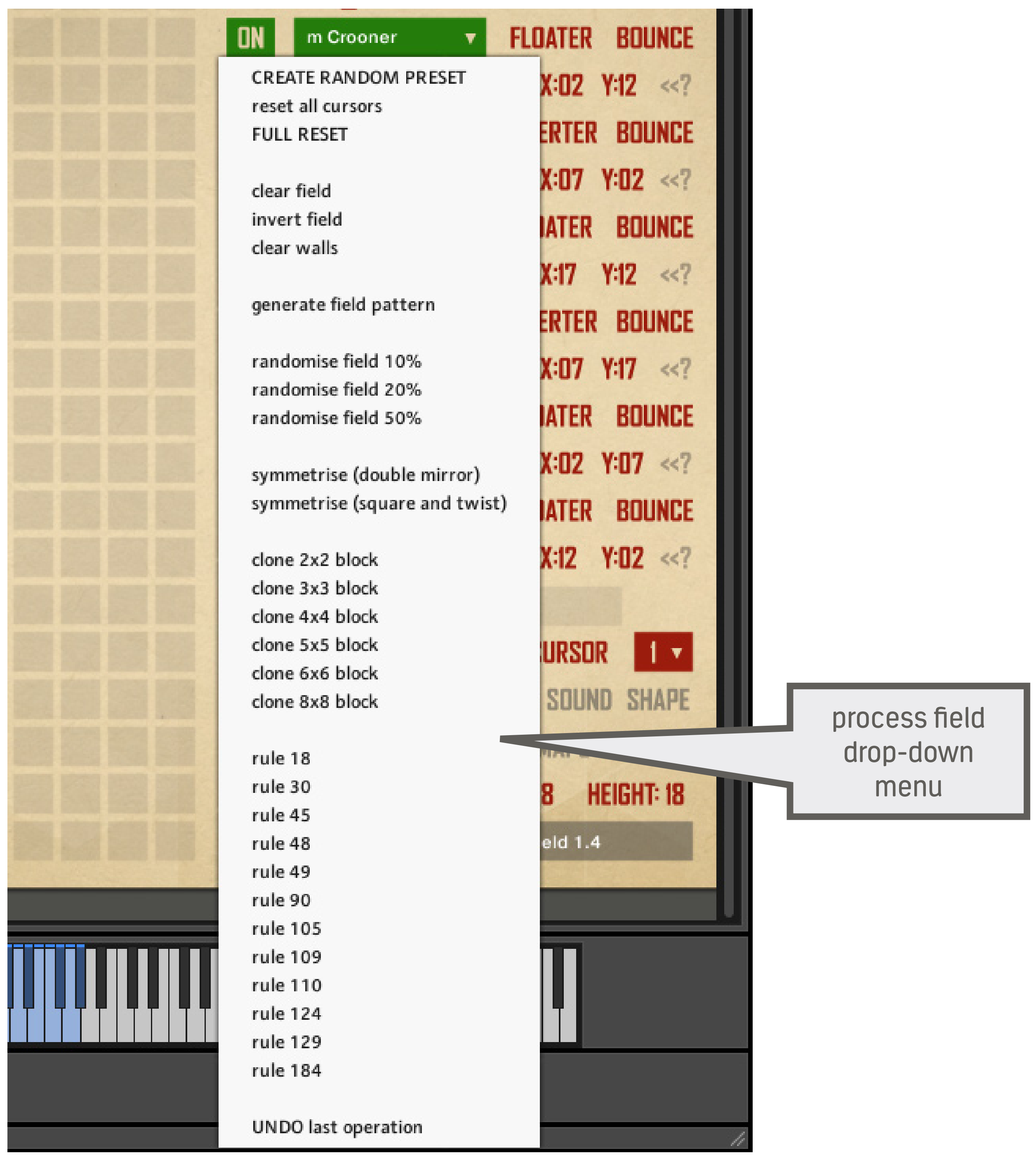
Symmetrise (square and twist) – clones upper-left quarter of the field. It is being rotated by 90 degrees and copied into following field quarters. Field width and height will be set to equal values. This option can be used to create "twister" programs (see page 13).

Clone block – pick a block of defined size from upper-left corner and fill the field with its copies.

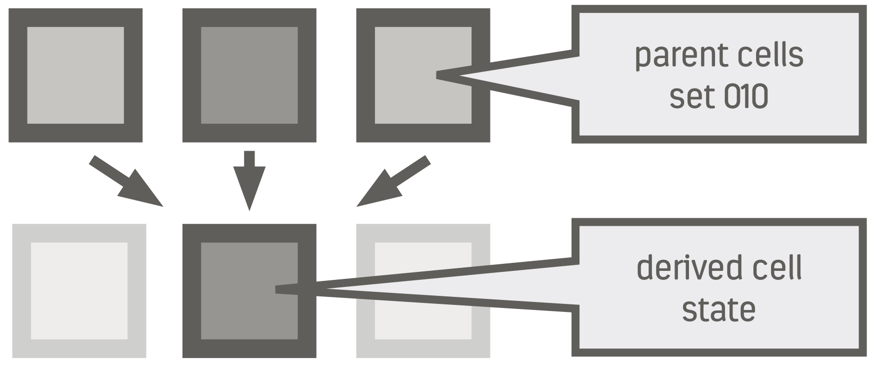
<u>Rule 30 and other rules:</u>

Following set of field generators use Stephen Wolfram binary cellular automaton rules, to generate field contents, using the top line as the seed. In this function state of each cell is derived from state of three cells directly above the cell being written. Rules define which set of cell states, produces which outcome. For example 111=0, 101=0 and so on, for each of 8 combinations.

Rule 30 states: 111=0, 110=0, 101=0, 100=1, 011=1, 010=1, 001=1, 000=0.
It can be simplified to binary string 0001110, which equals 30 in decimal system. Hence the name, Rule 30.



example of rule mechanics



Rule is being applied to following cells, row by row, downward the field. For a cell at the edge of the field, parent cell set is being wrapped around the field edge. So, for example upper left parent cell for cell 1,2 is cell 22,1.

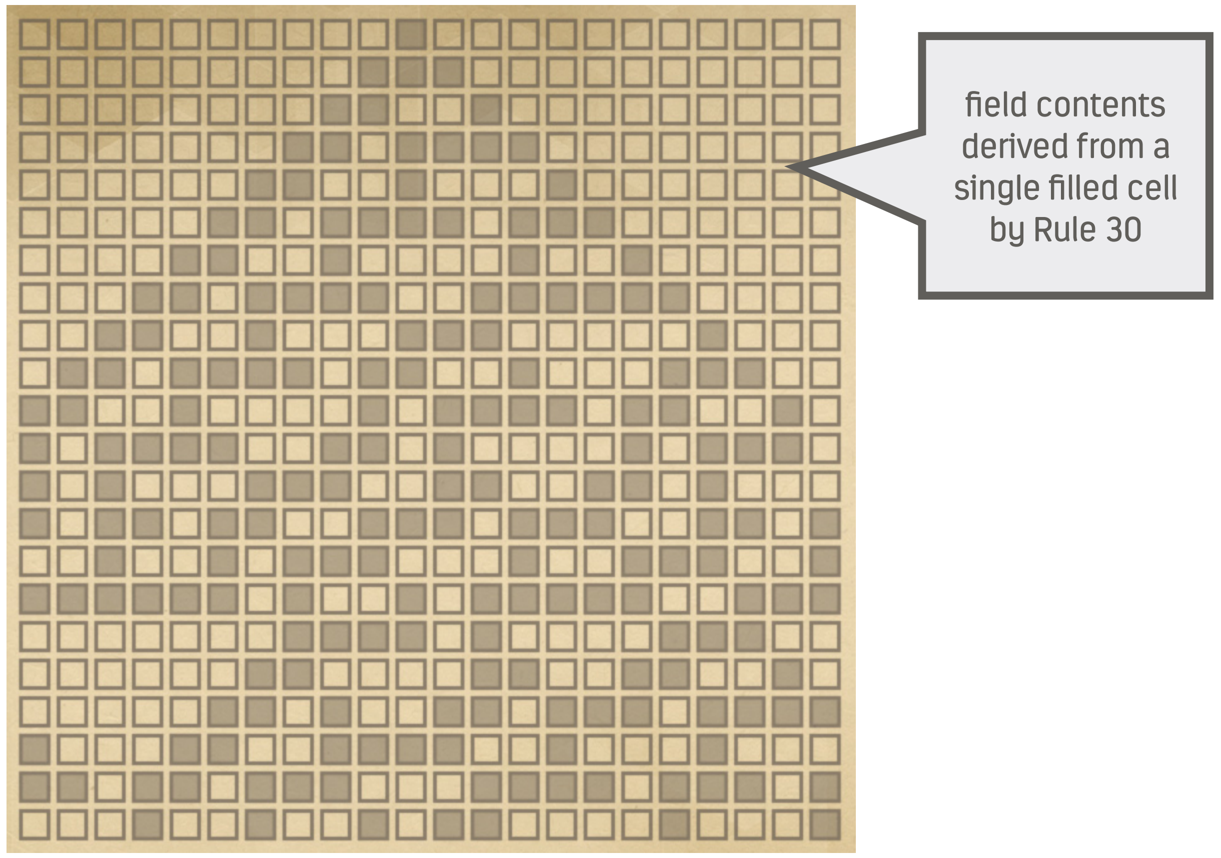
Selected field width is being used as pattern width, hidden field section to the right is not being written. On the other hand, the field is being written fully downward, including hidden part. That is, because the hidden part below the field, can be used to modulate field contents.

There are several rules to choose from, which produce different pattern types, being mixture of organic chaos and order. To use the rule based generator, program cells in the top line of the field and then apply a rule.

Note that this function needs a 'crackle' to produce something interesting, filling top line with uniform data (all empty cells, all filled cells or repeating "01" string), will generate a uniform field, likely a blank one.

The selection of rules is somewhat subjective, I picked the most known ones and ones, which I thought, can be useful in the context.

To learn more about the rules, see the Atlas of Elementary Cellular Automata at: http://atlas.wolfram.com/01/01/



<u>Remaining PROCESS FIELD menu options are not strictly field related:</u>

Create random preset – configure all sequencer options, filling it with more or less random data. The generator will set-up the field, cursors, note maps, occasionally it will use animator (but only move modes). When any melodic cursors are used it will also set up a chord progression, with exception for pentatonic scales which sometimes will have a progression and sometimes will not. Preset generator will respect preset loading filters (see page 42), you can turn off some parameter groups and only randomise some areas, for example note maps and progression.

Reset all cursors – clear setting for all cursors (the same can be achieved by resetting a cursor with EDIT ALL CURSORS switch on).

FULL RESET – resets all sequencer settings, including field and note maps.

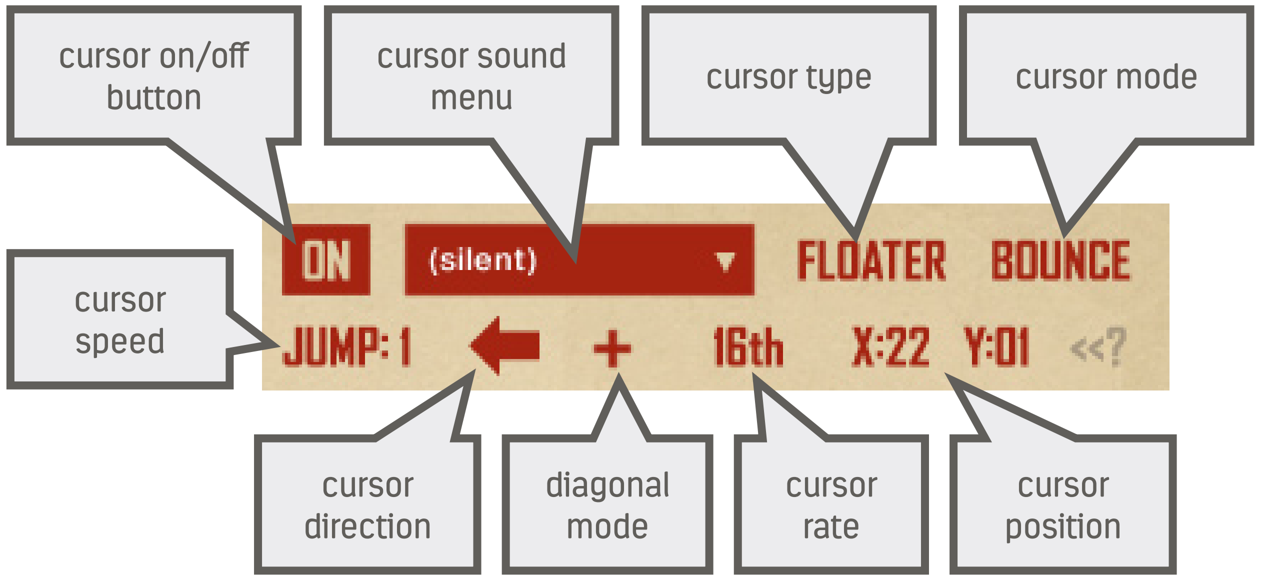
UNDO – cancel last operation. Undo works by writing a preset to a special slot, before performing certain operations. You can recall stored settings by selecting UNDO from the drop-down menu. If you're not happy with what has been restored, select UNDO again, to go back to state from before calling UNDO. Restore points are created for all mass operations: all options available in PROCESS FIELD menu, changes applied to all cursors, loading presets, etc. Restore points are not created for some functions which would be very easy to undo manually, like changing field dimensions. Undo function can be selective, recalling only defined data groups (see page 42).

Cursor mechanics and movement

Cracklefield can animate up to 8 cursors. Cursors can be used to trigger notes or just to interact with sequencer field and/or other cursors. Each cursor is displayed in different colour. The cell occupied by a cursor is coloured with the same colour as cursor on/off button and cursor sound menu.

Cursor can be turned on and off with ON button. An off cursor is not being displayed in the field and will be ignored when running the sequencer. Placing a cursor outside of defined field area (at one of unused cells), has the same effect as turning it off. Except, it's position is being displayed in the field grid.

Switch to MOTION tab to configure the way cursors move.



Cursor DIRECTION defines cursor heading (left/right/up/down). The controller acts as vertical slider.

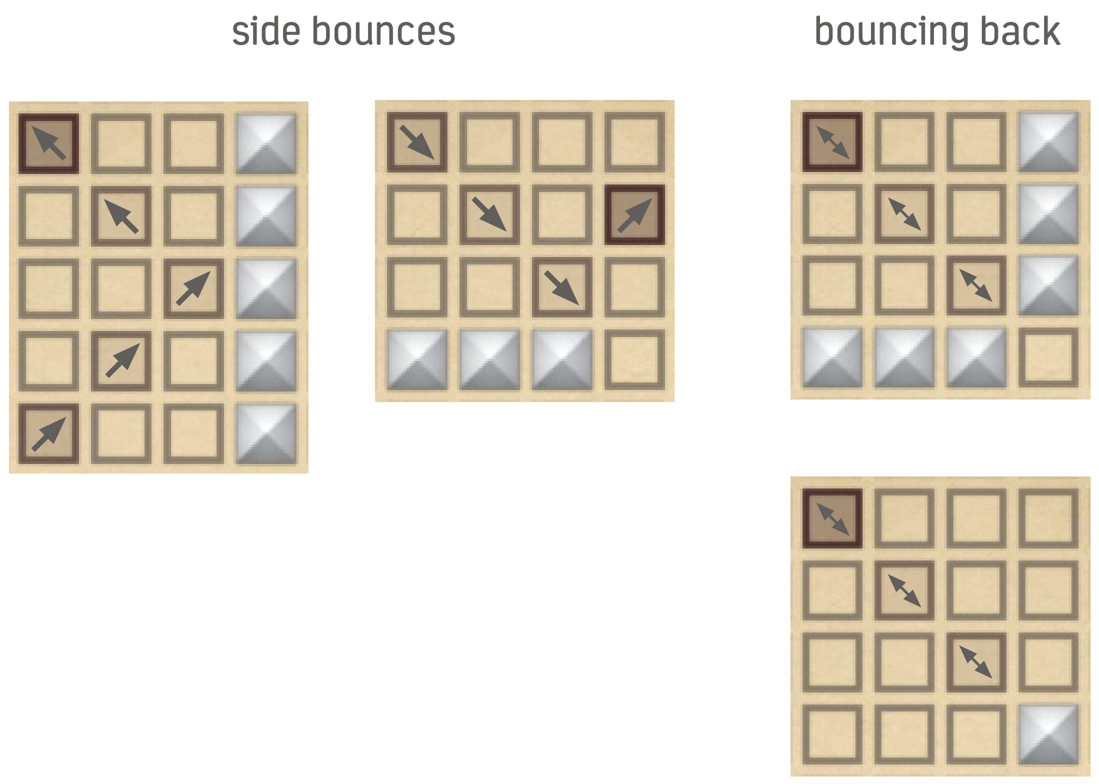
Diagonal mode switch (looking as "+" or "x", depending on setting), changes heading options to up-left, up-right, down-left, down-right. Switching cursor to diagonal mode, also changes the DIRECTION controller appearance.

There are two basic cursor modes, BOUNCE and PASS THROUGH.

In PASS THROUGH mode cursor behaves, as in regular sequencers. It moves forward until the field edge and then it wraps around reappearing on the opposite side of the field. In diagonal mode it reappears on side field edge, so it continues travelling in the same diagonal line. PASS THROUGH cursors do not interact with obstacles or other cursors, they treat wall cells as a gap in the field, teleporting instantly to the other side of a wall. This property can be used to create simple looped patterns of different length on the same field.

In BOUNCE MODE the cursor will bounce off field edges, walls and (if collision engine is enabled) other cursors. When running into an obstacle, bouncing cursor will change its direction and continue moving. In horizontal/vertical mode, the direction will be changed to opposite. Cursor travelling left will be travelling right, cursor travelling up will be travelling down. A collision cannot change cursor's direction from horizontal to vertical or the other way around, left-right/up-down movement can be only altered manually.

In diagonal mode bouncing mechanics are a bit more sophisticated, cursor can change direction by 90 degrees, or bounce back, depending on type of obstacle, as shown in illustration below.



Cursor POSITION controllers can be used to position the cursor at specified coordinates (however it is easier to achieve using SET CURSOR mode and clicking on desired cell). The X-Y controllers act as horizontal and vertical sliders and can be automated. They change values while the cursor is moving.

JUMP controller defines cursor speed, that is, how many cells the cursor travels in one sequencer step. If you set cursor speed to a value other than "1", it looks like cursor is jumping or skipping cells. The cursor will trigger sound depending on the value of the cell it moves onto during the step, however it will travel through all steps interacting with obstacles in the way. If you place a higher speed cursor against a single cell wall, it will not jump over, but it will bounce back. If COLLISIONS are enabled, it will also bounce off any cursors being in the way.

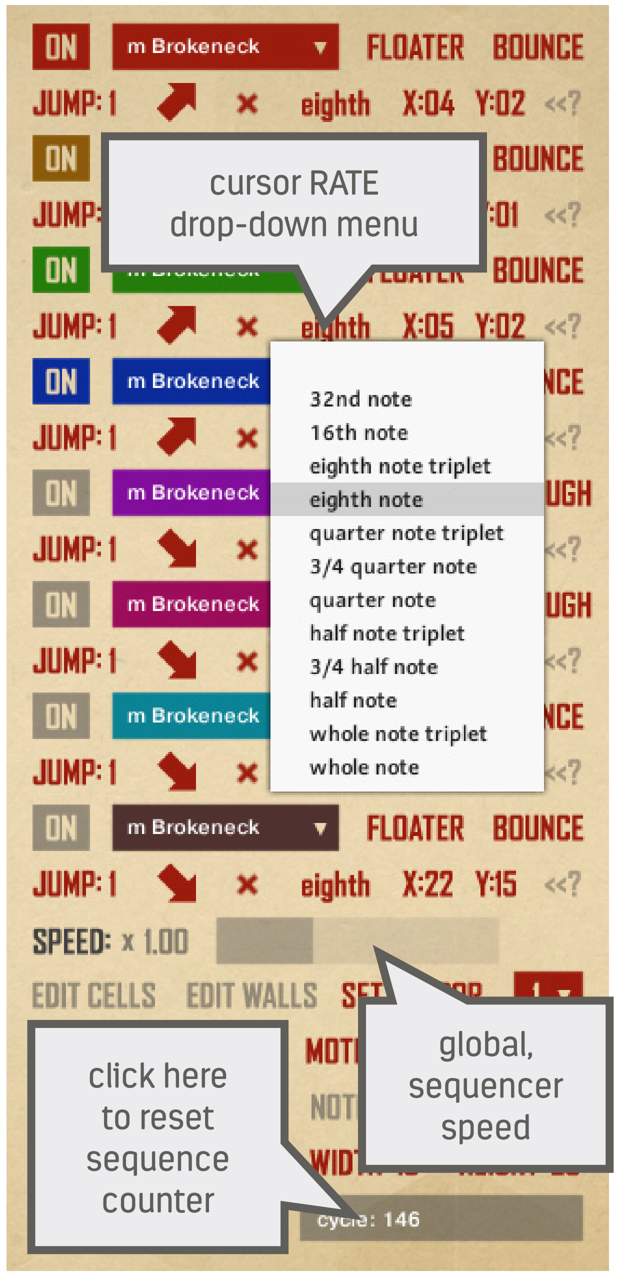
Cursor RATE, defines cursor movement timing, that is, how often and when cursor moves. Cursor rate is measured in note fractions (including triplets), so cursor movement is synchronized with host tempo. Smallest (fastest) available cursor rate is 32nd note, largest (slowest) rate is whole note.

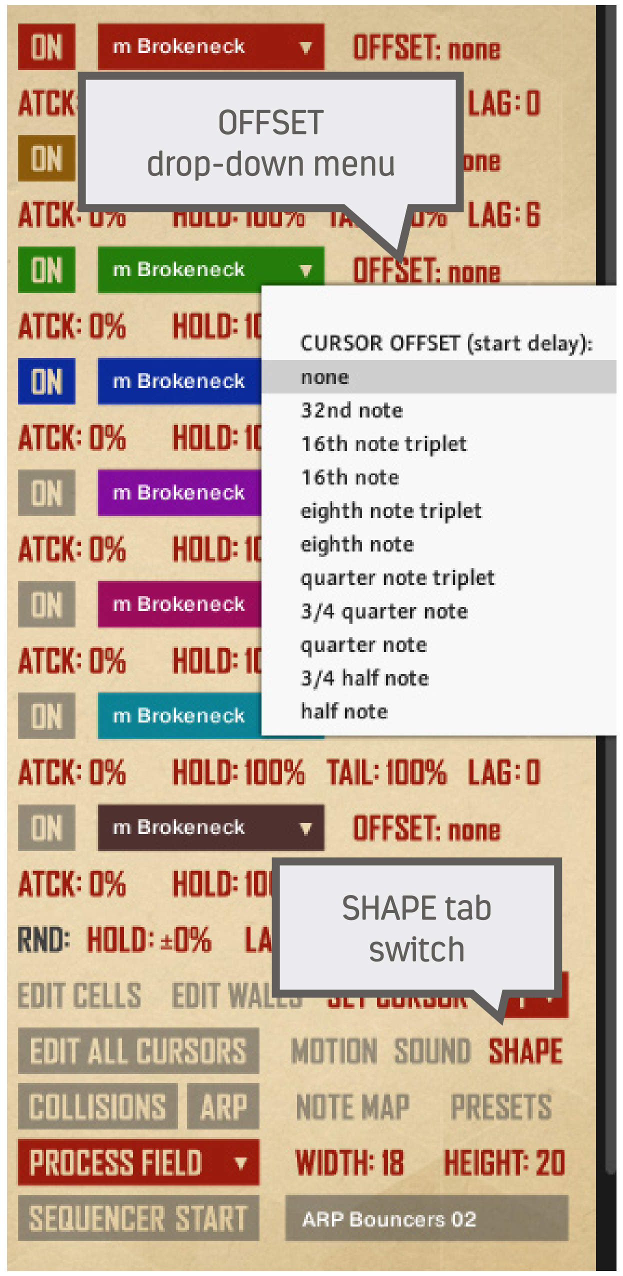
On the bottom of movement tab, there is sequencer SPEED controller. It can be used to adjust entire sequencer engine speed, relative to host's tempo. It can be adjusted, while sequencer is playing to create slowdown or speedup effect, or it can be pre-set to x0.5 or x2.0, which is equal to half tempo, or double tempo.

There is cursor parameter named OFFSET, which is related to cursor RATE. It is located in SHAPE tab. It will delay cursor movement relative to other cursors. It can be used to create a kind of interlaced cursor movement. For example, if you set two cursors to rate of eighth note, they will trigger notes simultaneously at every eight note. If you set one of those cursors to OFFSET of a 16th note, the cursors will not play simultaneously, as one of them moves a 16th note later than the other. The sequence would contain notes every 16th note and every second note would be generated by the other cursor.

Similarly you can set four cursors to rate of quarter note and set offset values to: none, 16th note, eighth note, 3/4 quarter note. Which is the reason for 3/4 quarter and 3/4 half to be included in rate values.

As you can see, cursors are moving at different time points. When the sequencer is running, it will update the sequence counter. Counter value equals to how many 32nd note steps have passed since the sequencer started. Starting/stopping the sequencer does not reset the counter, as it could change cursor's position in movement queue. This way you can pause/resume sequence, without changing the way it evolves. Sequencer counter is being reset when loading presets, when instrument is restarted/loaded, or it can be reset manually at any moment, by clicking on sequence counter label. Resetting the counter while the sequencer is running will make the sequence skip forward and will likely change cursor's queue order, if offset parameter is used.





Cursor collisions

As it has been mentioned, cursors can bounce off each other. Considering that there are up to eight cursors, which can have different rates, different speeds and move at different times, collision detector is not quite trivial function.

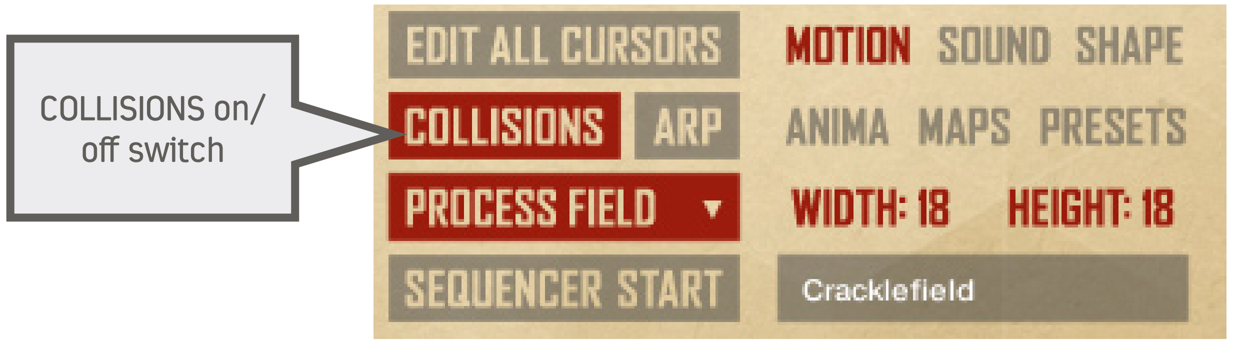
Cursor collision detector can be turned on/off using COLLISIONS switch. Only cursors in BOUNCE mode are analysed by the detector, cursors in PASS THROUGH mode are considered "invisible" to other cursors. So, if collision detector is off, bouncing cursors only bounce off static obstacles: walls or field edges. When COLLISIONS are on, they also interact with each other.

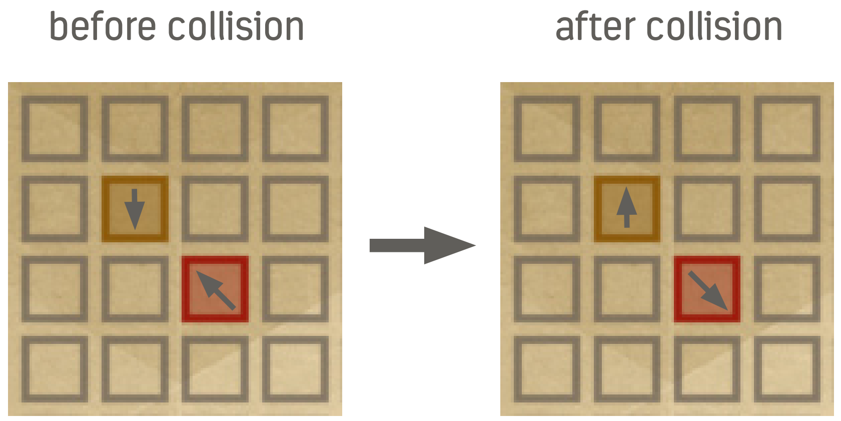
To make the collision detector relatively simple and fast, the cursors are being moved one after another, using a simple set of rules: a cursor cannot travel onto another cursor (pass through cursors excluded), if the way is clear, it moves, if there is another cursor in the way, cursor changes direction, as if it was bouncing off a static obstacle, additionally, it is changing direction of the other cursor (if the other cursor moves in collision course). If cursor is bounced, the detector leaves it alone and moves to next cursor. When it reaches last cursor, it starts from the beginning and tries to move cursors that haven't moved yet.

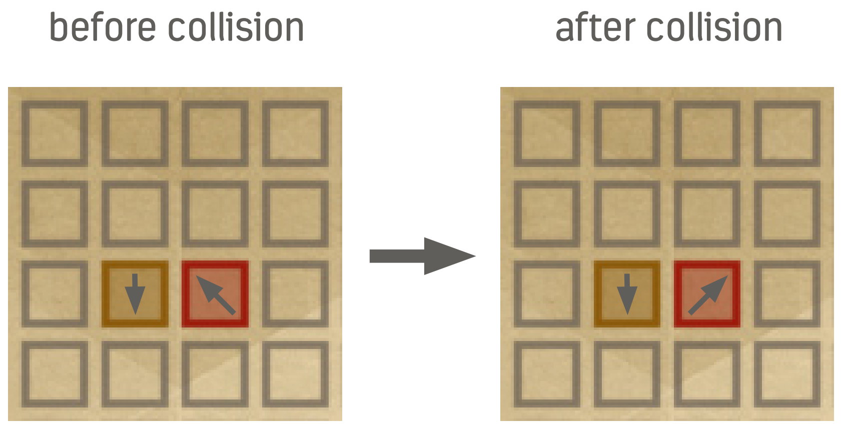
All bouncing cursors are analysed in a loop, until every cursor has found a clear cell to move to, or it has exceeded a defined number of cycles, then it is being considered "blocked" and left as is. If such event happens, the sequencer displays a message in Kontakt's message bar, stating "Unresolved collision for X cursors", where X is number of blocked cursors. Such situations may occur naturally depending on circumstances, for example when a fast rate cursor is blocked by several slower rate cursors waiting for their time to move. Easiest way to create a blocked cursor, is to surround it by walls (or slower rate cursors). In symmetric setups, it may happen that blocking a single cursor may destroy system's symmetry, which may depend on cursor order in the queue (as cursors are analysed in order).

It can happen that a moving cursor bounces off a cursor, which is not moving at given turn. In such situation waiting cursor changes direction, but doesn't move until its turn comes. Slow rate cursor can change direction several times, bounced by faster cursors, before it moves. It is possible that a fast cursor runs into the back of slower cursor travelling in the same direction, in which case, the faster cursor will bounce back and slower cursor will continue on its course.

If a diagonal cursor bounces off a horizontal/vertical cursor, it can change the other cursor's heading, but only, if it is on opposite course, as shown in illustration below.







Cursor types: floater and its variations

There are several cursor types to choose from. They differ, in the way cursor interacts with field contents, walls and field boundaries. The basic cursor type is named FLOATER, it has no impact on field contents or walls, it bounces off field edges and walls in the way described above. Next six cursor types are variations of a FLOATER, adding different kinds of interaction with field contents and walls.

<u>Cursor list:</u>

FLOATER – basic cursor type, no special properties.

INVERTER – flips the state of cells it travels onto; fills empty cells, clears filled cells.

PAINTER – fills every cell it travels onto.

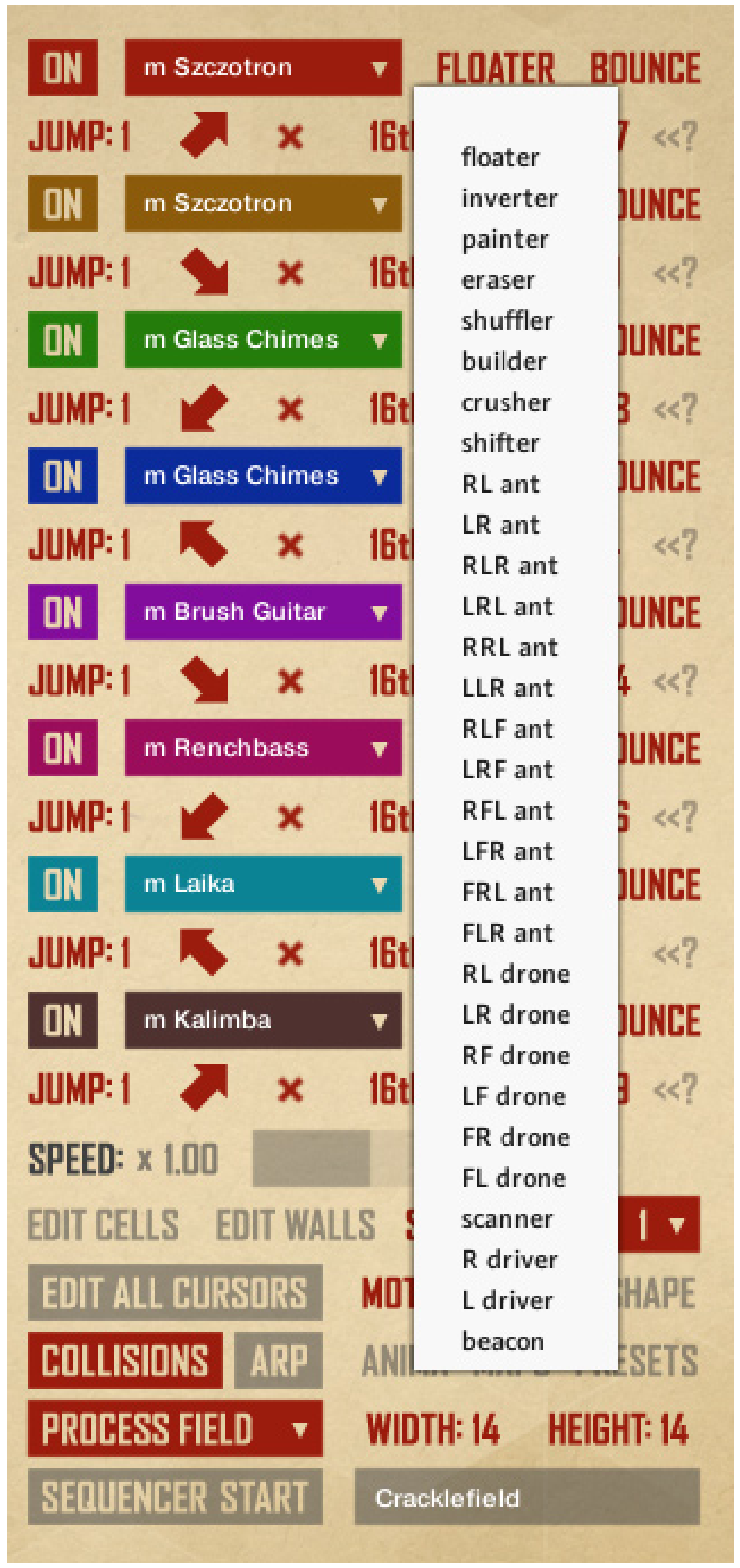
ERASER – clears every cell it travels onto.

SHUFFLER – is interesting one, it slides whole field row and/or column in the direction it travels. For example, if it is moving horizontally to the right, it slides the whole field contents row to the right. Moved data wraps around the field edge, the cells pushed beyond field edge reappear on the opposite side. In diagonal mode, it slides both, field row and field column. Note that in horizontal/vertical mode, it is quite useless as a playback cursor, because the cells are being moved along with it, so it either triggers a note on every step or never triggers one. It can be used to dynamically modify field contents, when set to silent mode.

Unlike previous cursors, SHUFFLER modifies the field in every discrete step, for example, if the cursor's JUMP setting is set to 3, the field row/column will be moved by 3 cells on each step. PAINTER OR ERASER, will only change the state of every 3rd cell, using the same setting.

Note that, except for shufflers, cursors trigger notes prior to changing field contents, they look if cell is filled or empty, play note on filled cell and then modify cell value.

cursor type drop-down menu



Following three cursor types interact with walls:

BUILDER – sets a wall on every cell it travels to. Builder doesn't detect walls, it doesn't bounce off walls and it doesn't teleport through walls. It still does bounce off cursors and field edges.

CRUSHER – in BOUNCE mode, it removes wall from a cell after bouncing off it. In PASS THROUGH mode it clears walls from cells it travels onto.

SHIFTER – similarly to SHUFFLER, it slides entire wall row/column. Similarly as a BUILDER, SHIFTER does not detect walls, it doesn't bounce off walls or teleport through. Note that collision detector will not check for collisions between moving walls and cursors, so a wall may be shifted onto a cursor.

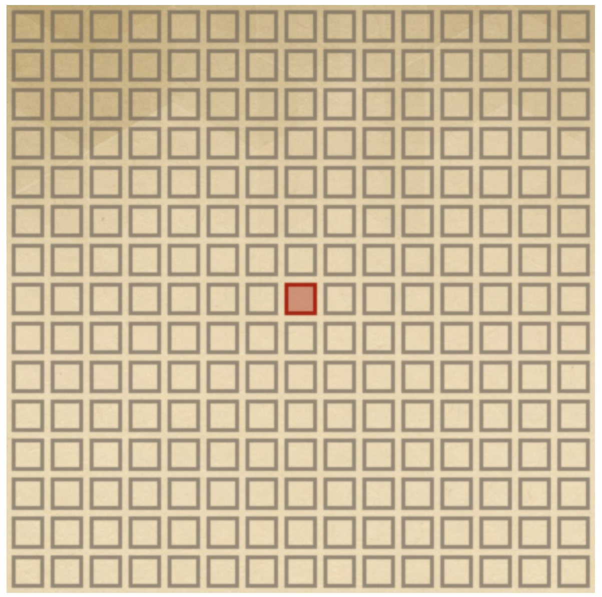
Cursor types: ants and drones

Following family of cursors is based on mathematical phenomenon named "Langton's ant". It is two dimensional cellular automaton, with very simple program, which produces surprisingly complex patterns.

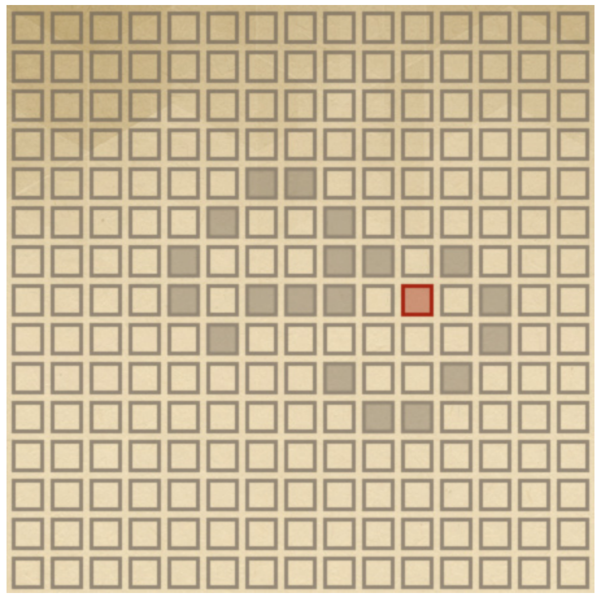
The basic ant program is:
At empty cell, turn 90 degrees right, change cell state, move forward.
At filled cell, turn 90 degrees left, change cell state, move forward.

Example pattern development for single ant is illustrated below.

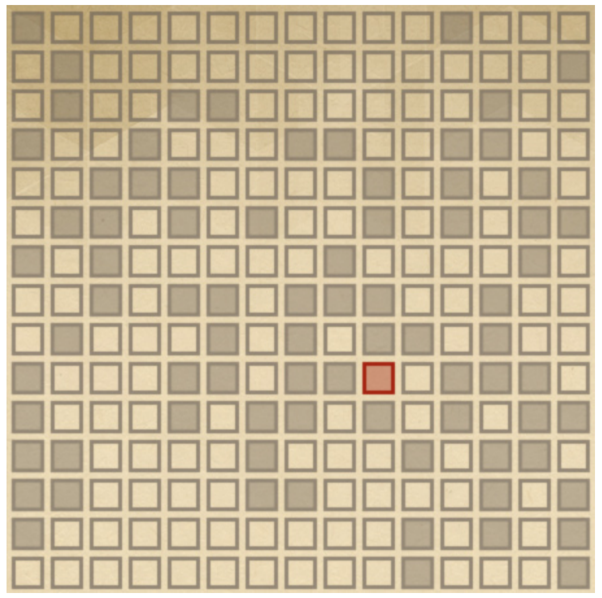
initial pattern



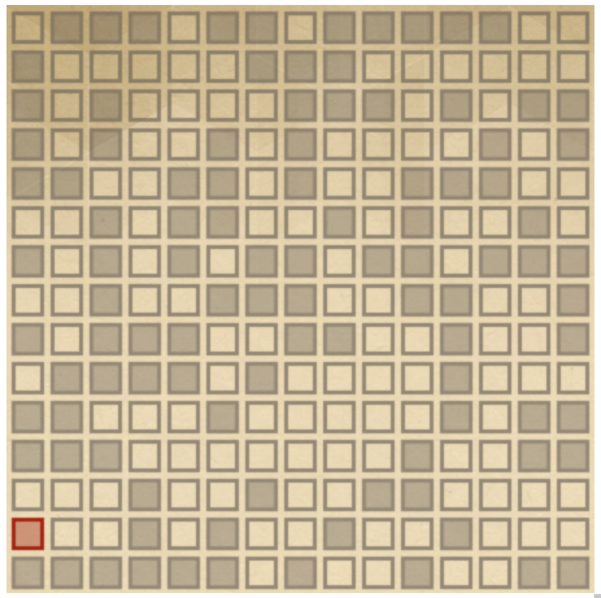
pattern after 100 steps



pattern after 1000 steps



pattern after 5000 steps



Ants are fascinating, as it feels like observing some basic forces of nature at play. With Cracklefield you can also listen to them.

Ant cursors behave like regular ones, they bounce off each other, interact with walls and field edges. The difference is that they change visited cell state and they change direction depending on cell value.

Basic ant cursor types are:

RL Ant – which stands for right-left, turn right at empty cell, turn left at filled cell.

LR Ant – left-right, turn left at empty cell, turn right at filled cell.

There is extended set of ant types, based on idea of having more than two cell states. Here three states are being used. Cells with value of "0" are empty cells. Value of "1" equals a filled cell. For the third state, the instrument uses cells with value of "2", just for three state ants.

Every other cursor will recognize a "2" cell as an empty cell. It is also not possible to program those cells manually. A cell can be given value of "2" only by a three step ant cursor. When visited by such cursor a "0" becomes "1" cell, "1" cell becomes "2" cell and a "2" cell becomes "0" cell. Two step ants treat "2" cell as an empty cell, so they change "2" cell to "1" cell.

Three state ants are named by their program, just as basic ants.

For example, RRL Ant, means: at cell with value of "0" turn right, at cell with value of "1" turn right, at cell with value of "2" turn left.

Additionally there's third program command, "F", as forward, which equals to 'do not change direction'.

Program for RFL Ant, decodes as: at cell with value of "0" turn right, at cell with value of "1" do not change direction, at cell with value of "2" turn left.

Here's list of available three state ants:

RLR Ant – right-left-right
LRL Ant – left-right-left
RRL Ant – right-right-left
LLR Ant – left-left-right
RLF Ant – right-left-forward
LRF Ant – left-right-forward
RFL Ant – right-forward-left
LFR Ant – left-forward-right
FRL Ant – forward-right-left
FLR Ant – forward-left-right

Another set of ant derivative cursors, is called DRONES. A drone is an ant which does not change cell state, it changes direction depending on cell value, but it does not interfere with it. Drones are not particularly useful alone, but they can create interesting setups when combined with other cursors, following maps creates by ants or bounce off each other in a prepared field. Only two state drones are available.

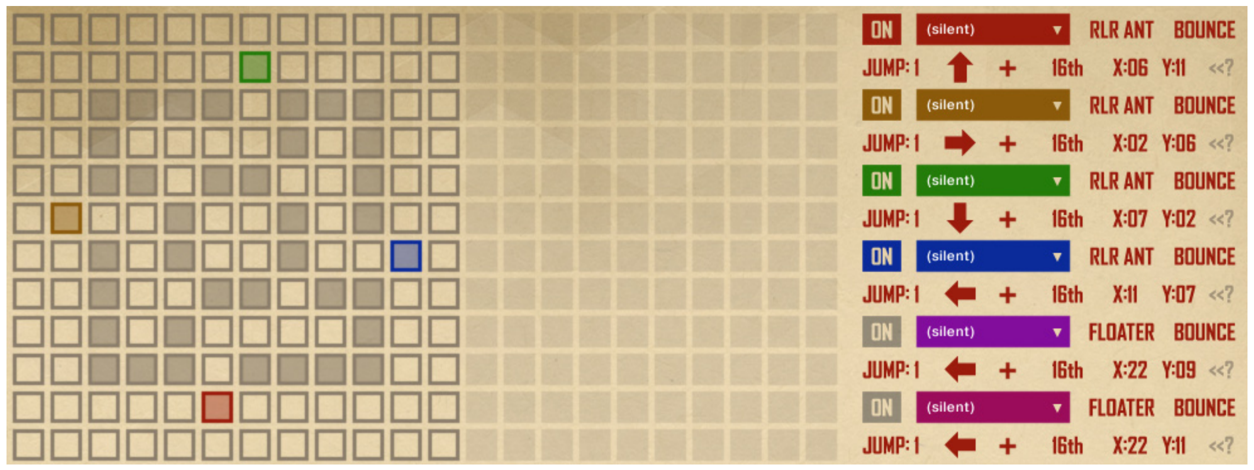
List of available drones:

RL Drone – right-left
LR Drone – left-right
RF Drone – right-forward
LF Drone – left-right
FR Drone – forward-right
FL Drone – forward-left

There is interesting type of ant cursor setup, called 'twister' in preset list. It's a kind of symmetrical setup where you have four ants of the same type, set as you had taken a quarter of the field and copied it to other three quarters, rotating field contents, cursor position and direction by 90 degrees for each following quarter. In such setup all cursors move symmetrically, creating a kind of kaleidoscope effect.

Cracklefield provides shortcuts for creating such setups. First one is "symmetrise (square and twist)" option in PROCESS FIELD menu. It will prepare the field contents (alternatively you can clear the field and let cursors create the pattern). Note that both field dimensions should be equal, for this setup to work. Second shortcut is available in cursor randomisation menu (signed "<<?") for first and fifth cursor, named "symmetrise 4 cursors (twister)". This function will clone given cursor properties to next three cursors, adjusting cursors position and direction.

example of pattern generated by a twister setup



```
ON  (silent)  RLR ANT  BOUNCE
JUMP: 1  ↑  +  16th  X:06  Y:11  <<?
ON  (silent)  RLR ANT  BOUNCE
JUMP: 1  →  +  16th  X:02  Y:06  <<?
ON  (silent)  RLR ANT  BOUNCE
JUMP: 1  ↓  +  16th  X:07  Y:02  <<?
ON  (silent)  RLR ANT  BOUNCE
JUMP: 1  ←  +  16th  X:11  Y:07  <<?
ON  (silent)  FLOATER  BOUNCE
JUMP: 1  ←  +  16th  X:22  Y:09  <<?
ON  (silent)  FLOATER  BOUNCE
JUMP: 1  ←  +  16th  X:22  Y:11  <<?
```

Cursor types: scanners and drivers

There are three additional special movement mode cursors.

SCANNER – moves like a floater, but when it runs into an obstacle, it makes additional move, one step down. It is handy for testing field maps, as it attempts to visit every cell in the field.

L DRIVER – when it runs into an obstacle, it turns 90 degrees left. When left alone in the field it will ride around its edges. In BOUNCE mode it turns when running into walls, cursors or field edges. In PASS THROUGH mode it only turns at field edges.

R DRIVER – same as L DRIVER, but it turns right.

Note that all three cursors only work in horizontal/vertical mode.
In diagonal mode they move as a FLOATER.

Cursor types: beacon

BEACON cursor is designed to work with field animators. It doesn't move by itself or change field contents. When it is placed on a filled cell it plays a note as any other cursor. It can be used to create a setup where the field content moves and cursors remain static.

When a BEACON is in BOUNCE mode it will change position when hit by another (moving) cursor. Diagonally moving cursors will only push BEACON when it's hit "in the corner". When such cursors collide being "side by side", BEACON will remain where it was and the other cursor will bounce off.

Configuring cursor sound

Next to cursor on/off switch, there is drop-down menu for selecting a sample set assigned to given cursor. A cursor can be used to play melodies or percussive patterns. There are several sample sets to choose from, melodic sets are prefixed with "m", percussive with "p".

Custom sample sets can be easily added (see page 40).

Click on SOUND tab to display sound related controllers.

The most important setting is CURSOR MAPPING MODE, selected from drop-down menu. It determines the algorithm for deciding which notes to play and when. First three options, USE MAP with variations, are designed for creating melodies. Next four modes are for use with percussive instruments. Mapping modes will be explained in following sections.

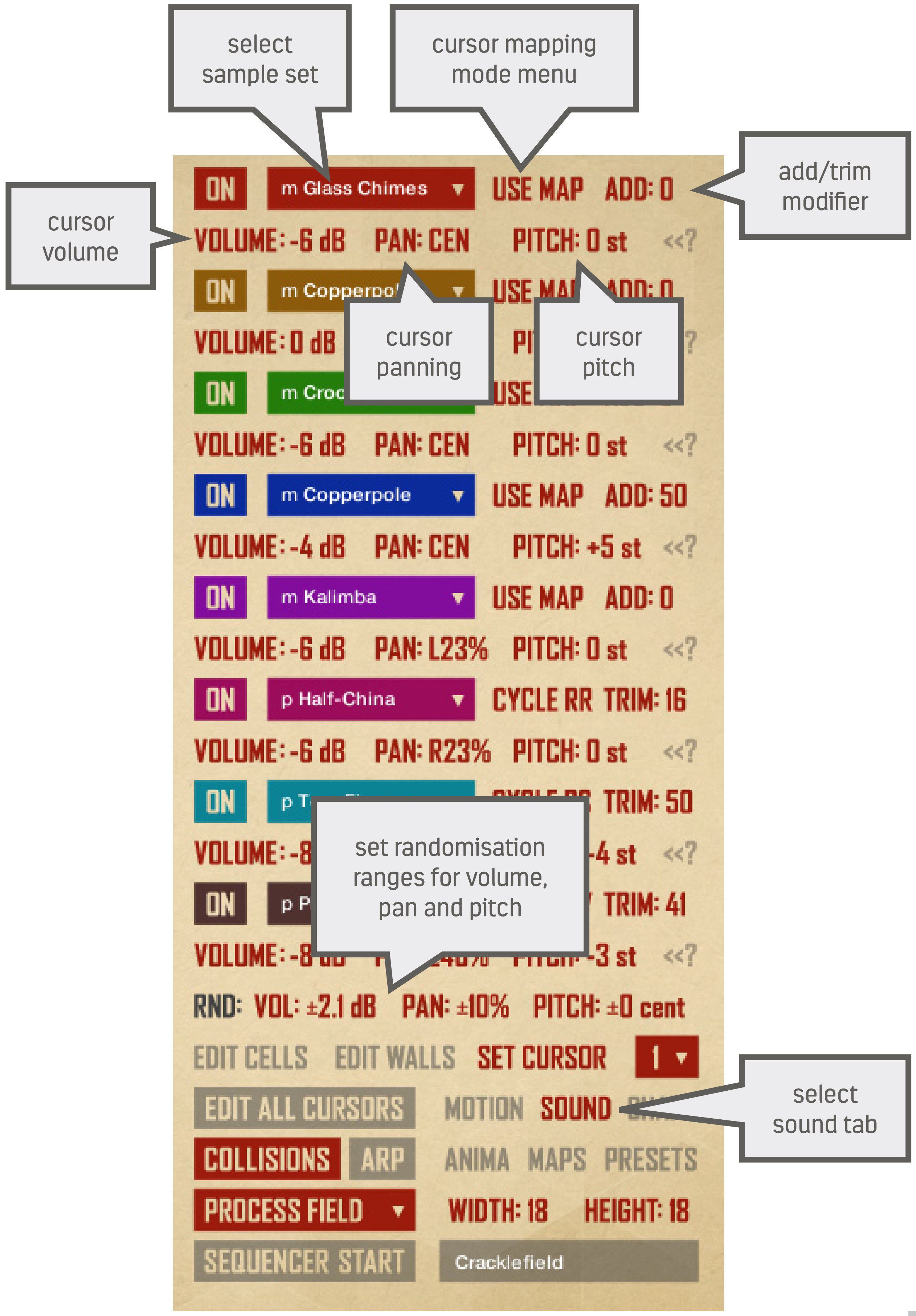
ADD/TRIM controller is modifier for mapping mode, it has different function for different mapping modes, although they all share the controller value.

Then there are controllers for VOLUME, PAN and PITCH for each cursor. These parameters are applied per note and are assigned to cursor and not to a sound. So you can have different volume, pan and pitch settings for each cursor, even if they are all assigned to the same sample set.

Below the cursors, there are controllers for VOLUME, PITCH and PAN randomisation. Set the range to value other than zero to make the sequencer apply random parameter change for each note it generates.

ADD/TRIM, VOLUME, PAN and PITCH controllers act as vertical sliders.

Note that you can play a sample set by hand, using midi keyboard. The played sample set will be one assigned to currently selected cursor (use drop-down menu next to SET CURSOR switch, to select a cursor).



ON m Glass Chimes USE MAP ADD: 0
VOLUME: -6 dB PAN: CEN PITCH: 0 st <<?
ON m Copperpol USE MAP ADD: 0
VOLUME: 0 dB
ON m Croc USE
VOLUME: -6 dB PAN: CEN PITCH: 0 st <<?
ON m Copperpole USE MAP ADD: 50
VOLUME: -4 dB PAN: CEN PITCH: +5 st <<?
ON m Kalimba USE MAP ADD: 0
VOLUME: -6 dB PAN: L23% PITCH: 0 st <<?
ON p Half-China CYCLE RR TRIM: 16
VOLUME: -6 dB PAN: R23% PITCH: 0 st <<?
ON p T TRIM: 50
VOLUME: -8 -4 st <<?
ON p P TRIM: 41
VOLUME: -8 -3 st <<?
RND: VOL: ±2.1 dB PAN: ±10% PITCH: ±0 cent
SET CURSOR 1
WIDTH: 18 HEIGHT: 18
Cracklefield

Note map

For playing melodic content, the sequencer is using the note map, that is, an array which assigns a specific note number to each cell. When a cursor travels onto a filled cell, it reads the note number assigned to given cell and plays that note. To use note map, pick USE NOTE MAP mode from cursor mapping menu.

Note map is being created automatically according to given parameters, it can also be modified manually, if you wish to do so.

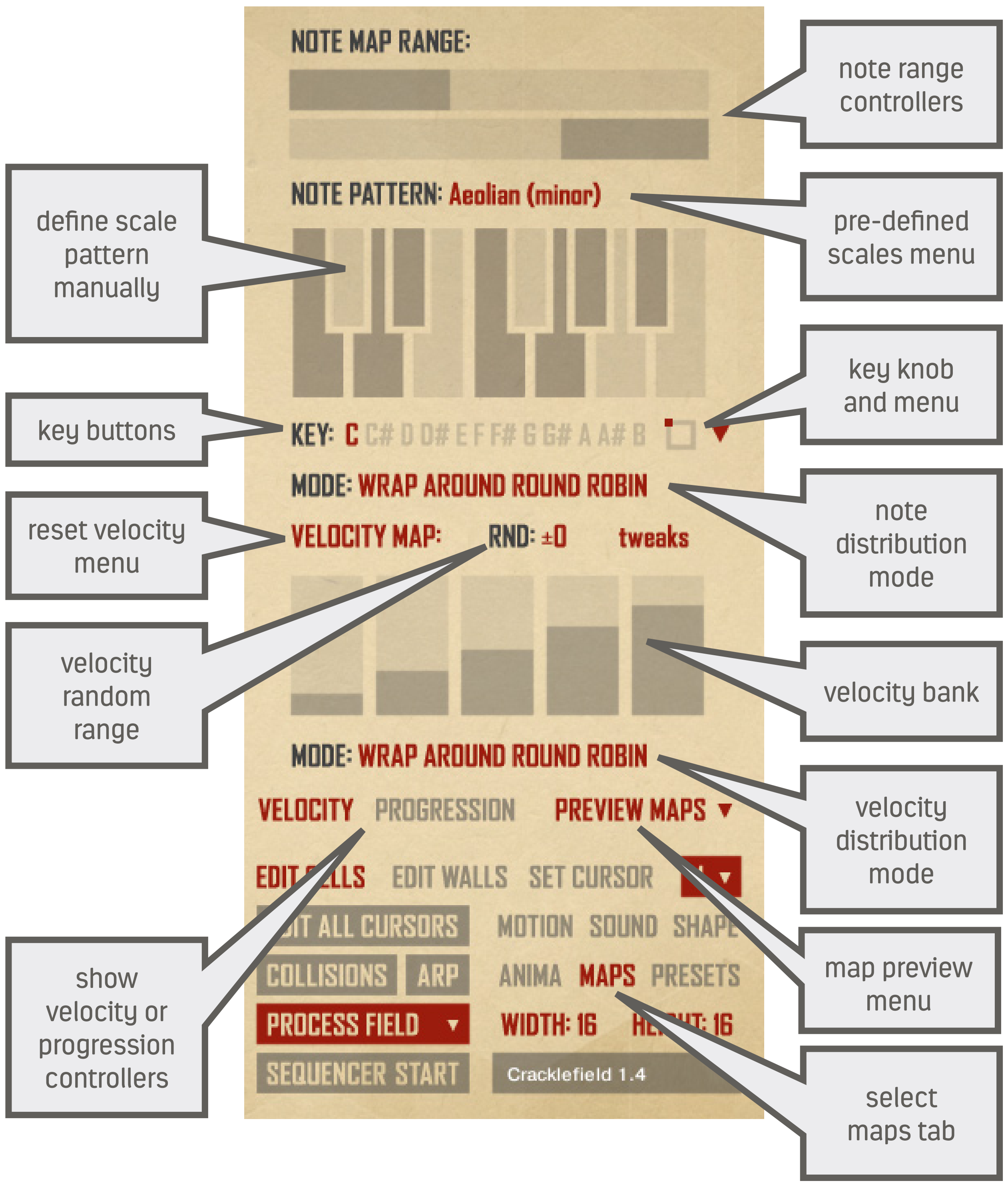
Click on MAPS tab, to display note mapping parameters.

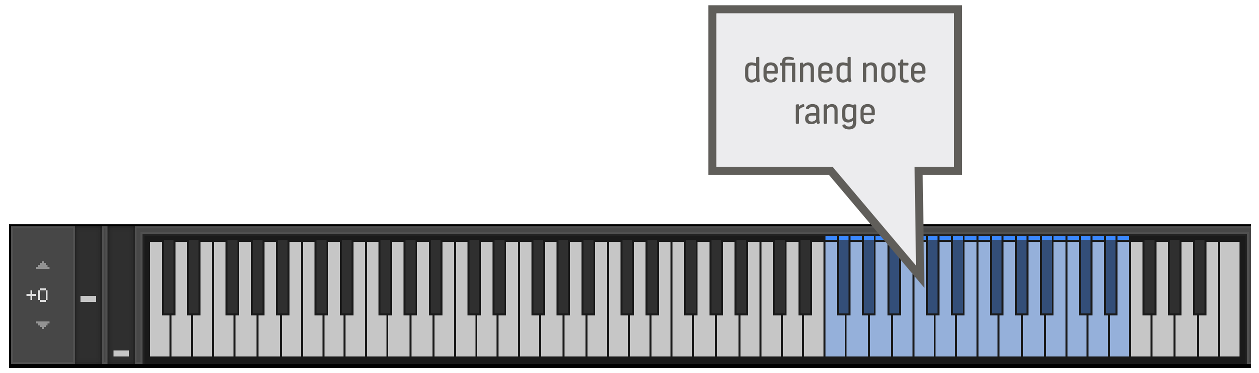
Note map will be created according to RANGE, NOTE PATTERN (SCALE), KEY and NOTE DISTRIBUTION algorithm.

Set NOTE RANGE sliders to define lowest and highest notes to use for the map. Defined range will be displayed in blue colour on virtual keyboard. Only notes from defined range will be used for filling the map. Note that the range only determines which notes will be played by the sequencer, the whole keyboard range can be played manually with midi keyboard (in Cracklefield sample sets are mapped to the whole note range, white keys do not indicate an "empty" key here).

NOTE PATTERN indicates which notes in an octave should be used. It is equivalent of picking a scale. You can pick one of pre-defined patterns, named according to scale names they represent, from a drop down menu (click on note pattern name).

NOTE PATTERN can be also programmed manually, by clicking on keys in pattern controller. Note pattern name will then be set to "custom". You can check, if manually programmed pattern matches any of pre-defined ones. Pick the last option from pattern menu: "(match pattern and key)". The program will search pattern bank and set pattern name and key, if it finds a match. Otherwise it will display message "no match for current pattern".


NOTE MAP RANGE:
NOTE PATTERN: Aeolian (minor)
KEY: C C# D D# E F F# G G# A A# B
MODE: WRAP AROUND ROUND ROBIN
VELOCITY MAP: RND: ±0 tweaks
MODE: WRAP AROUND ROUND ROBIN
VELOCITY PROGRESSION PREVIEW MAPS
EDIT CELLS EDIT WALLS SET CURSOR
EDIT ALL CURSORS MOTION SOUND SHAPE
COLLISIONS ARP ANIMA MAPS PRESETS
PROCESS FIELD WIDTH: 16 HEIGHT: 16
SEQUENCER START Cracklefield 1.4
note range controllers
pre-defined scales menu
define scale pattern manually
key knob and menu
key buttons
reset velocity menu
note distribution mode
velocity random range
velocity bank
velocity distribution mode
show velocity or progression controllers
map preview menu
select maps tab



You can change pattern 'key' using one of KEY buttons. Changing KEY will slide defined pattern accordingly (for example, changing key from C to C# will slide entire pattern one step to the right). For custom defined patterns, the program assumes the user defined a pattern that matches currently selected key.

Alternatively you can change key using a knob placed next to KEY buttons set. The knob is there to keep compatibility with earlier versions of Cracklefield, where primary KEY controller was a slider which could be automated. Key can be also selected from a drop-down menu, which again is a comes from earlier version. Additionally KEY can be selected using defined key-switches on MIDI keyboard, so it can be easily changed while playing (see page 41).

Select a note distribution MODE to set note mapping mode – define how note pool is being distributed to specific cells. There are 6 available modes:

WRAP AROUND ROUND ROBIN – assigns following notes to following cells, when the highest note is reached, it starts from the lowest note.

WRAP AROUND UP AND DOWN – assigns following notes to following cells, when the highest note is reached, it proceeds downward until reaching the lowest note and so on.

INTERLACED 2 STEP – assigns following notes skipping every second note, up and down, for each new line it starts from the beginning, increasing start point by 1. Creates more evenly distributed map.

INTERLACED 3 STEP – same as 2 step, but skips every three notes.

OCTAVES HORIZONTAL (ROUND ROBIN) – each field line contains notes from one octave in round robin, next line contain notes from next octave. After reaching last octave, it starts from the beginning.

OCTAVES HORIZONTAL (SPREAD) – each field line contains notes from one octave, octaves are being spread evenly. For example, if they are only two octaves first 11 lines will contain first octave, next 11 lines will contain second octave.

Here are note distribution algorithm examples, assuming the note range spans for two octaves, only available notes in the pattern are C, D and E and the field dimensions are 5x5 (in fact they are always 22x22, for writing a note map, field width and height controllers are not considered).

WRAP AROUND ROUND ROBIN

C1	D1	E1	C2	D2
E2	C1	D1	E1	C2
D2	E2	C1	D1	E1
C2	D2	E2	C1	D1
E1	C2	E2	D2	C1

WRAP AROUND UP AND DOWN

C1	D1	E1	C2	D2
E2	D2	C2	E1	D1
C1	D1	E1	C2	D2
E2	D2	C2	E1	D1
C1	D1	E1	C2	D2

OCTAVES HORIZONTAL (RR)

C1	D1	E1	C1	D1
C2	D2	E2	C2	D2
C1	D1	E1	C1	D1
C2	D2	E2	C2	D2
C1	D1	E1	C1	D1

OCTAVES HORIZONTAL (SPREAD)

C1	D1	E1	C1	D1
C1	D1	E1	C1	D1
C1	D1	E1	C1	D1
C2	D2	E2	C2	D2
C2	D2	E2	C2	D2

It's best to pick distribution mode by ear, try different modes to see which works best in given setup.

If you'd like to check current map exact assignments, use MAP PREVIEW menu. Pick note numbers or note names, to display note assignments over field cells.

You can tweak cursor generated notes further, by changing MAPPING MODE, ADD modifier and cursor PITCH.

ADD parameter changes the cell index position used to read mapped note number, making cursor fetch notes from different map area. For example, cursor position is x=2, y=1 and add=2, the sequencer looks at cell 2,1 to see if the cell state is empty of filled, if it's filled, it looks at map coordinates 4,1 to fetch note number to play (2 is added to cell index just for the purpose of reading the note map). Cell index numbers go from left to right downward. So first field line covers number from 0 to 21, next from 22 to 43 and so on. You can preview index numbers using MAP PREVIEW MENU.

Again, it's best to approach ADD parameter by ear, tweak it, to get different note patterns from the same cursor route.

Additionally, there's REVERSED mapping mode, which makes note map to be read in reverse, first cell is assigned last value in the map and so on (just for purpose of reading the note map).

In USE MAP mode, changing cursor PITCH does not simply detune played sound, but it adds to note number. The note number is then being aligned to nearest note matching defined note pattern, so cursor always plays "in tune". In other words, there is "force to scale" tuning function which is always active in USE NOTE MAP mode. If you do not want to use tuning, pick chromatic scale as note pattern. Using PITCH parameter can make the instrument play notes out of defined range.

Chord mode

Third tuned mapping variation is named USE NOTE MAP (CHORD). It is simple chord generator, making a single cursor play several notes at once at a filled cell.

When CHORD mode is selected, ADD parameter is being transformed to chord size modifier. It displays as DYAD, TRIAD, TETRAD, PENTAD, HEXAD, HEPTAD and OCTAD, that is, 2-note chord, 3-note chord and so on up to 8-note chord.

Notes in a chord are calculated by walking two steps forward through defined note pattern or playing following notes in pattern when progression generator is active.

For example, let's assume note pattern matches major scale (ionian) in C (white keys). Note pattern is C-D-E-F-G-A-B.

Now, if mapped note is C1, a DYAD chord will be C1-E1, TRIAD will be C1-E1-G1, TETRAD C1-E1-G1-B1 and so on. Every other note in the pattern is skipped.

Cursor PITCH is used to determine first note in the chord, remaining notes are calculated from the first one.

Using LAG parameter in CHORD mode will delay each note in the chord, creating a strumming effect (see page 27). The strummed chords will play upward or downward, depending on cursor position.

Progression generator

Cracklefield can generate chord/pattern progressions. You can create a set of sub-patterns/chords and then switch through individual patterns as the sequencer is playing. A new note map will be generated each time a new chord is selected. Chords can be loaded manually or sequenced.

To activate the progression generator, click the PROGRESSION button. You will notice that the notes of the selected chord will be coloured red on the note pattern keyboard.

There are 12 editable chords which can be organised in a sequence of up to 36 steps. The sequence is split into three pages with 12 steps viewable on each page. When the SEQUENCE button is on, progression steps will automatically be selected when the main sequencer is running.

You can select a sequence step manually by clicking on one of the small empty buttons in the bottom row of the sequence programmer. Any of 12 chords can be assigned to a given step using drop-down menus – rectangular controllers in the middle row of the sequence programmer.

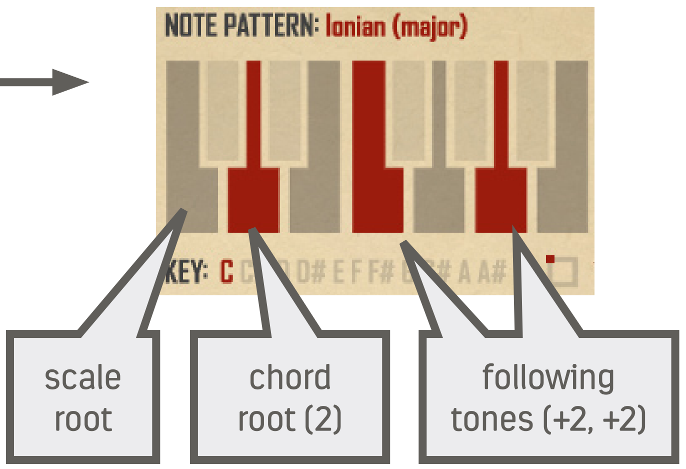
Chords are named using the numbers 1–12. Pre-defined chords follow the formula: root = chord number, +2, +2. So, for example, in chord 1, the first note is pattern root note (main pattern key), next note is two tones away from first note (counting only notes that belong to selected scale) and third note is two tones away from the second note.

Since chords are defined in such a relativistic way, they will automatically translate to any scale you select. For example in Ionian C:

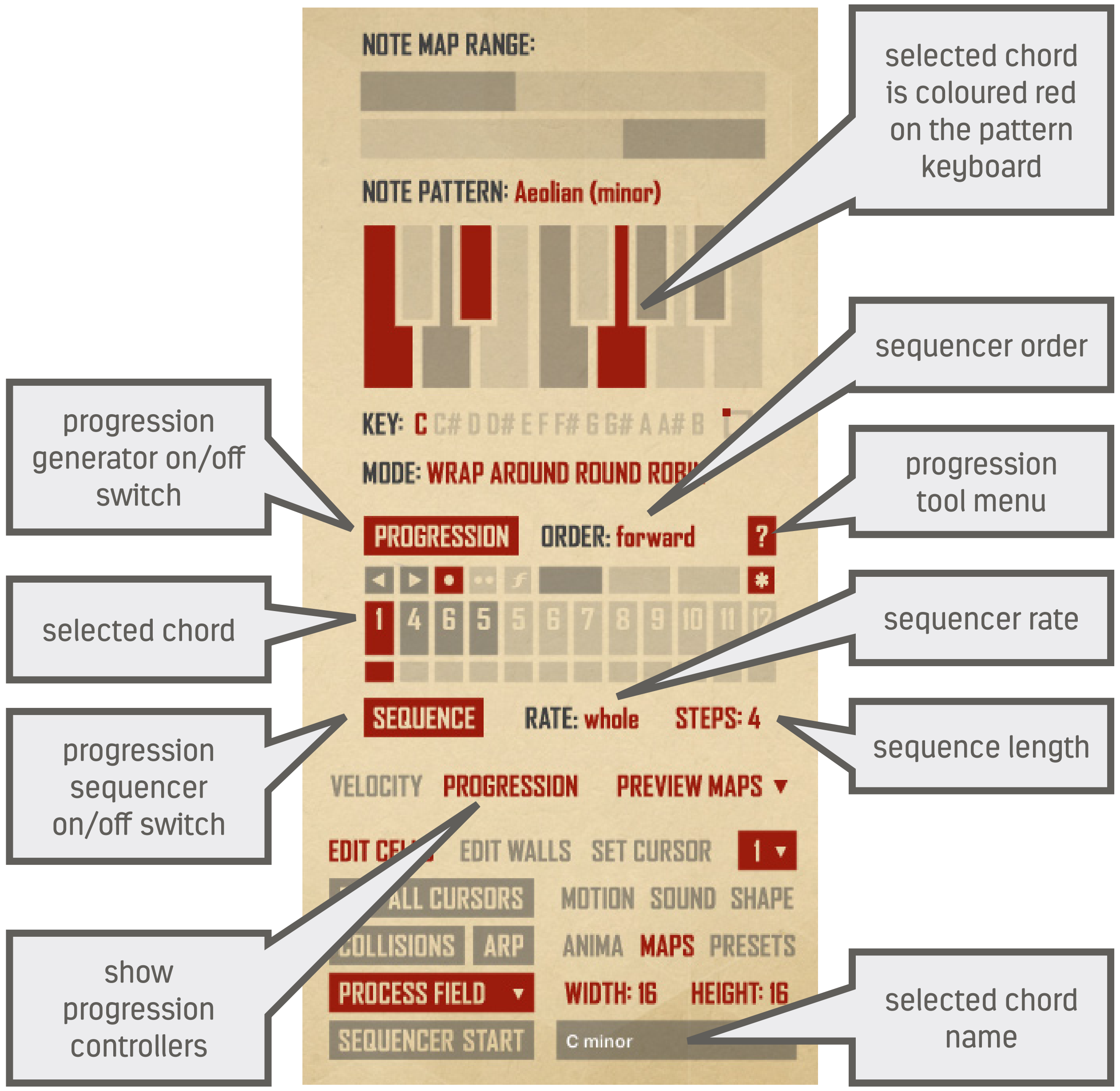
chord 1 = C-E-G = C major = I
chord 2 = D-F-A = D minor = ii
and so on

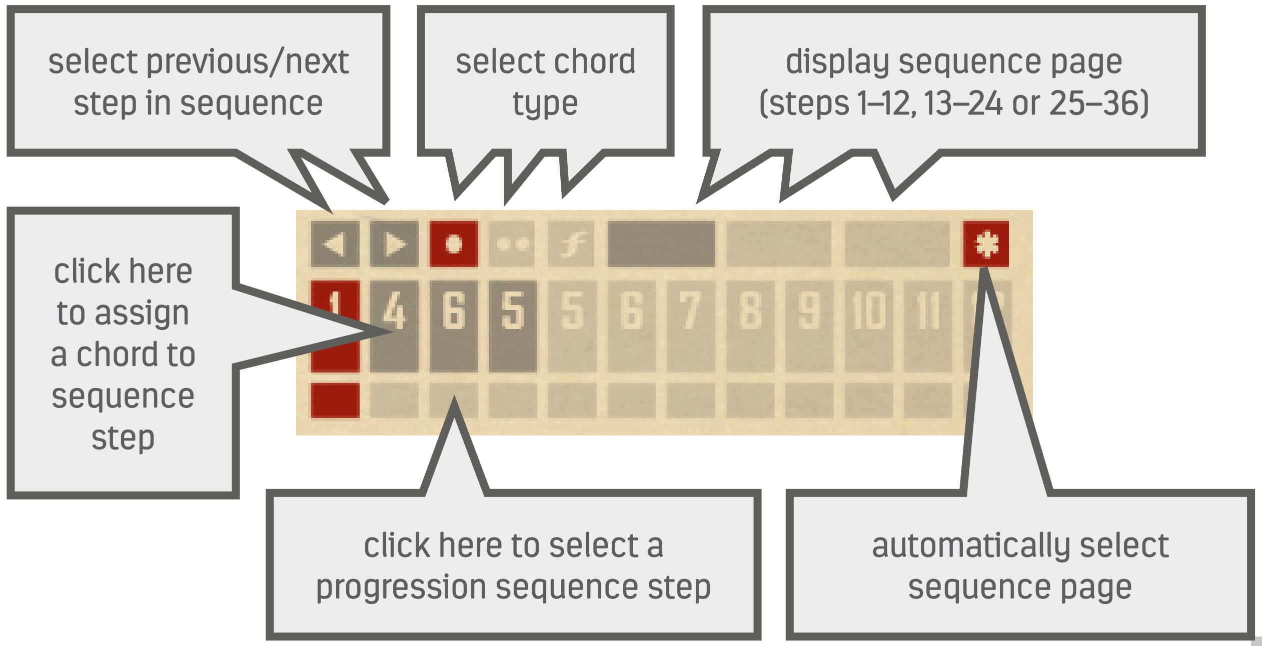
same chord formula applied to Aeolian E translates to:

chord 1 = E-G-B = E minor = i
chord 2 = F#-A-C = F# dim = ii^{0}
and so on



NOTE PATTERN: Ionian (major)
KEY: C C# D D# E F F# G G# A A# B
scale root
chord root (2)
following tones (+2, +2)

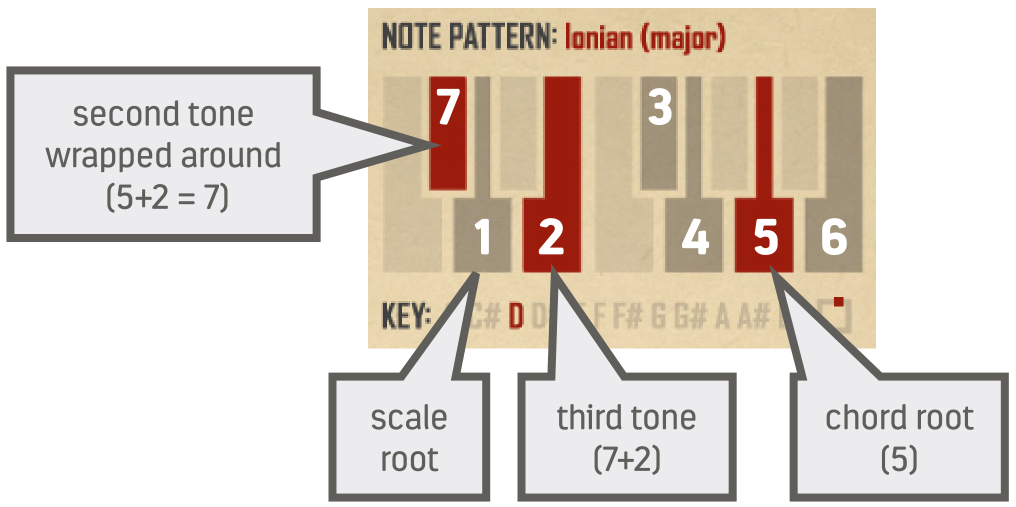




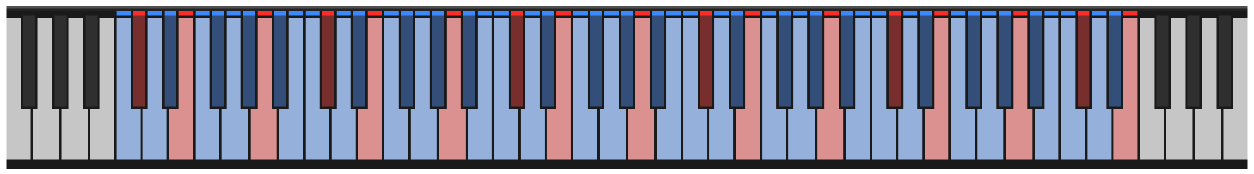
select previous/next step in sequence
select chord type
display sequence page (steps 1–12, 13–24 or 25–36)
1 4 6 5 5 6 7 8 9 10 11
click here to assign a chord to sequence step
click here to select a progression sequence step
automatically select sequence page

The chord formula is applied over a 12 tone pattern in wrap-around mode, as in the example below:

Ionian D – chord 5 (A major)


NOTE PATTERN: Ionian (major)
second tone wrapped around (5+2 = 7)
7
3
1
2
4
5
6
KEY:
scale root
third tone (7+2)
chord root (5)

Selected chord mask will be applied over defined note range by repeating the 12 tone mask. Notes that belong to selected chord mask will be coloured red on virtual keyboard, as illustrated below.



For a seven tone pattern/scale, chord 8 is the same as chord 1 (because root 8 is trimmed to fit 7 note pattern, becoming 1), chord 9 is the same as chord 2 and so on.

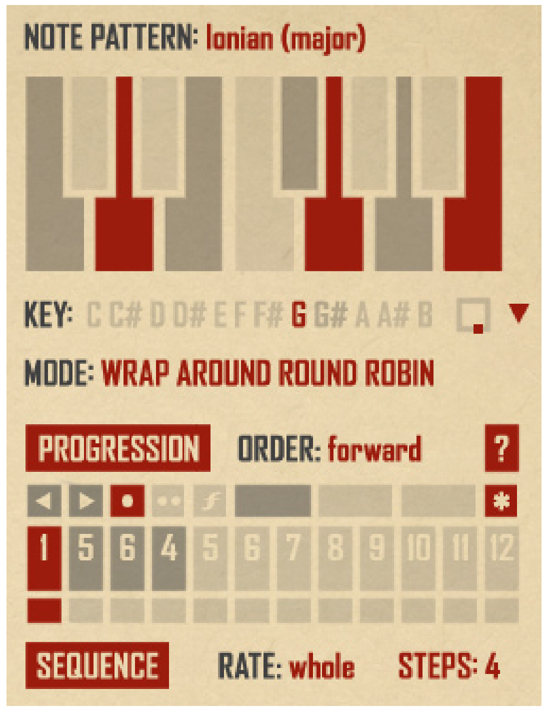
Cracklefield will recognise basic chord types and display selected chord name in the sequence counter box (when maps tab is selected). If a chord is not recognised, it will display the note sequence of the chord. Because chords are applied in wrap-around manner, the machine will not make distinctions between inversions.

Progression sequence

Use STEPS controller to set progression sequence length. Unused steps will be displayed in a lighter shade of grey. Turn on SEQUENCE button and the next progression step will be automatically selected as the main sequencer is running. Use RATE controller to define how often to change steps. Alternatively you can move the sequence manually – leave SEQUENCE button off and use next/previous buttons to select steps (small arrow buttons above first and second step). Or you could just automate step selection buttons, in which case sequence length controller has no use, as it only defines sequence border step.

STEPS controller may not be very convenient to use; alternatively, hold ALT key and click on step selection button to define new last step in the sequence.

Example: to set up popular I–V–vi–IV progression, set sequence length to 4, program sequence to 1, 5, 6, 4 and select Ionian (major) pattern scale. If you switch the pattern to Aeolian (minor), the progression will translate to i-v-VI-iv.



NOTE PATTERN: Ionian (major)

KEY: C C# D D# E F F# G G# A A# B

MODE: WRAP AROUND ROUND ROBIN

PROGRESSION ORDER: forward

1 5 6 4 5 6 7 8 9 10 11 12

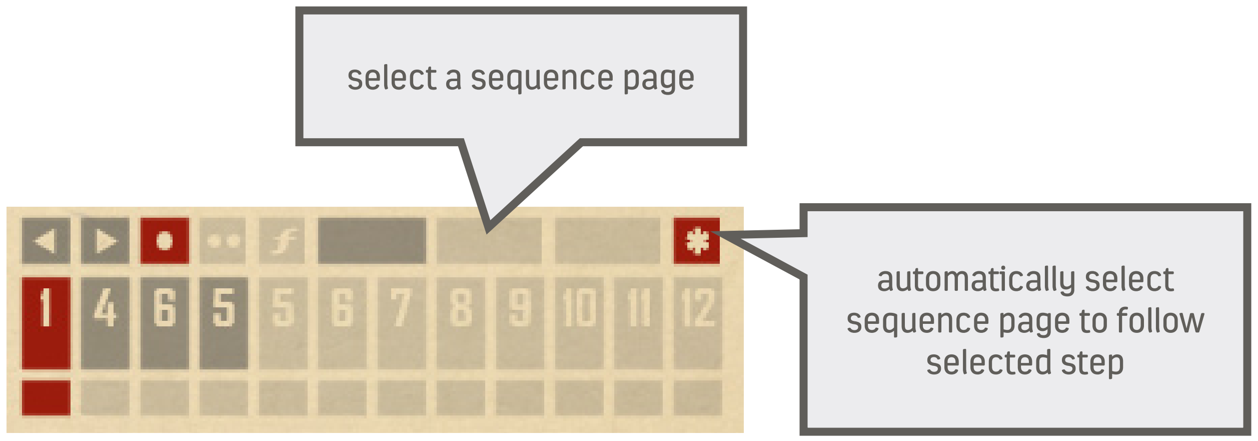
SEQUENCE RATE: whole STEPS: 4

When using pattern sequencer, the machine will select following steps according to ORDER controller. The most simple option is 'forward', in which case example sequence will play as 1-5-6-4-1-5-6-4... It can be changed to 'ping-pong' and it will go forward and backward: 1-5-6-4-6-5-1-5...

Both modes are synchronised with main sequencer clock. If you stop and start the sequencer, progression sequence will continue from where it has stopped. To restart the sequence, you need to restart the main sequencer clock – click on sequence counter box, next to main sequencer START button. Sequencer always restarts when loading presets.

There are three more sequence ORDER modes: random, auto (by cursors) and auto (by field). In random mode, next sequence step is being selected randomly from steps within range of sequence length. Auto modes are somewhat similar to random mode, but number of new step is synchronised with situation on the field. For auto (by field) mode, at the moment of step change, the machine will calculate a checksum of the field contents (cells being on or off in defined field boundaries). Then the checksum will be used to draw a number. The idea is that the same field map will always generate the same draw, so if you are using a static (not changing) field, the progression sequence will not be moving – so it's a good idea to only use 'by field' option when the field is evolving (for example ants or animator are used). 'By cursors' mode works similarly, but instead of the field checksum, a checksum of cursor configuration is being calculated. Cursor checksum depends on positions of all cursors on the field. Also different cursors have different 'weight', so the cursor order matters (unused cursors are not counted).

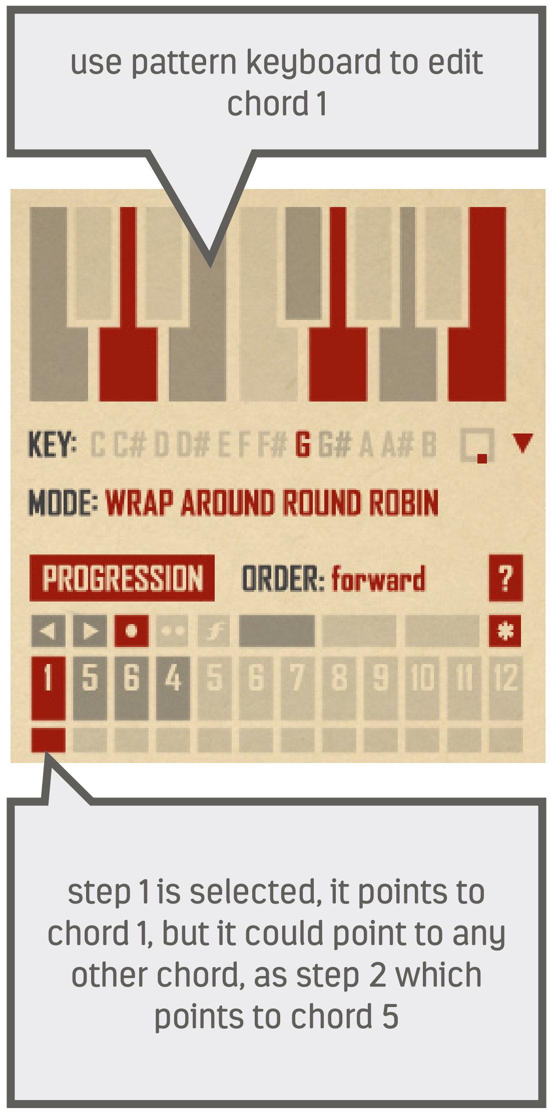
As mentioned earlier, a sequence can be up to 36 steps long with the sequence controllers being split into 3 pages of 12 steps. Use one of three wide blank buttons to select a page. When a small star/ asterisk button is enabled, whenever the sequencer selects a new step outside of the currently displayed page, the appropriate page will automatically be selected; otherwise the newly selected step will not be visible.



Programming chords

Each of 12 available chords can be programmed manually, so you are not restricted to standard triad chords. To edit a chord, first select a sequence step which points directly to one of 12 chords (other sequence program options will be explained in the next section, page 22). Now simply click on the note pattern keyboard to turn notes on or off. Note that you can only select notes that belong to the currently selected scale. Also, you can't have less than 1 note in a 'chord' and you can't select more than 7 notes.

A changed chord is saved instantly. When you have several sequence steps pointing to chord 1, reprogramming the chord will affect all those steps. Programmed chords are saved within presets. To reset all chords to default, use the progression tool menu (question mark box) and pick 'reset progression patterns'.


use pattern keyboard to edit chord 1
KEY: C C# D D# E F F# G G# A A# B
MODE: WRAP AROUND ROUND ROBIN
PROGRESSION
ORDER: forward
?
1 5 6 4 5 6 7 8 9 10 11 12
step 1 is selected, it points to chord 1, but it could point to any other chord, as step 2 which points to chord 5

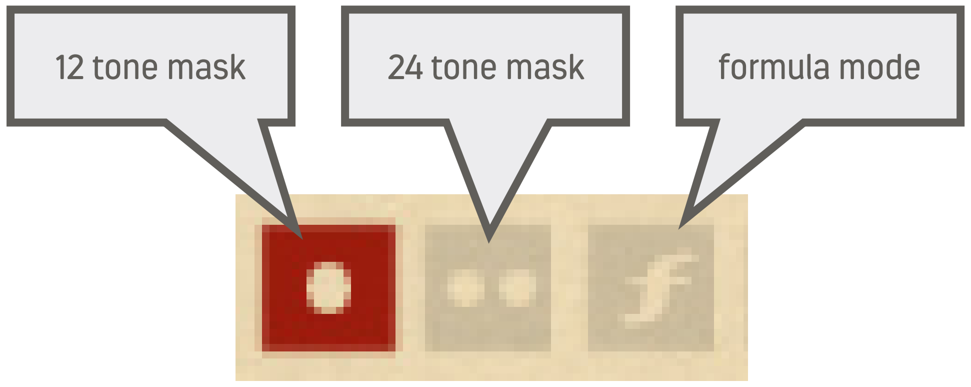
The pattern keyboard works s a chord programmer when PROGRESSION switch is on. To program a custom scale, turn the PROGRESSION button off.

Manually-programmed chords can be useful when you want to use a progression which does not fit a single scale and key, like A minor - A major. In such a case, you can select a chromatic scale and define the following chords manually (for example, chord 1 = A minor, chord 2 = A major).

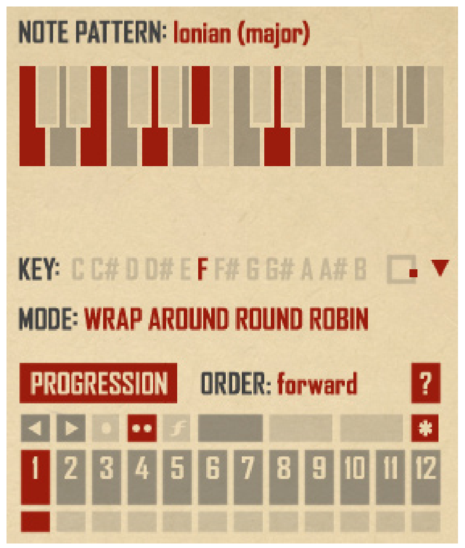
Note that the programmed chord is automatically being converted to the relativistic formula (root position + tone intervals within scale) and is stored as such. So if you select a different scale, the defined chords will automatically translate to fit the new pattern.

Chord modes

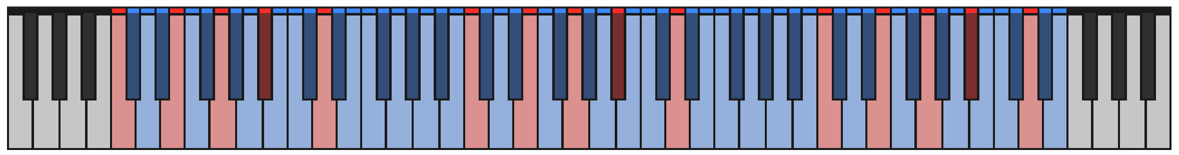
Each of 12 available chords can be set to one of three modes. The basic default mode is 12 tone mask, which is then applied to all octaves in defined note range. It is simple to program and may be good enough for many users, although it has some obvious limitations. For example, it is impossible to program 9th or 13th chords. For more advanced chord programming there are two more chord modes available: 24 tone mask and formula mode. When a sequence step pointing directly to a chord is selected, you can switch selected chord's mode using mode buttons.



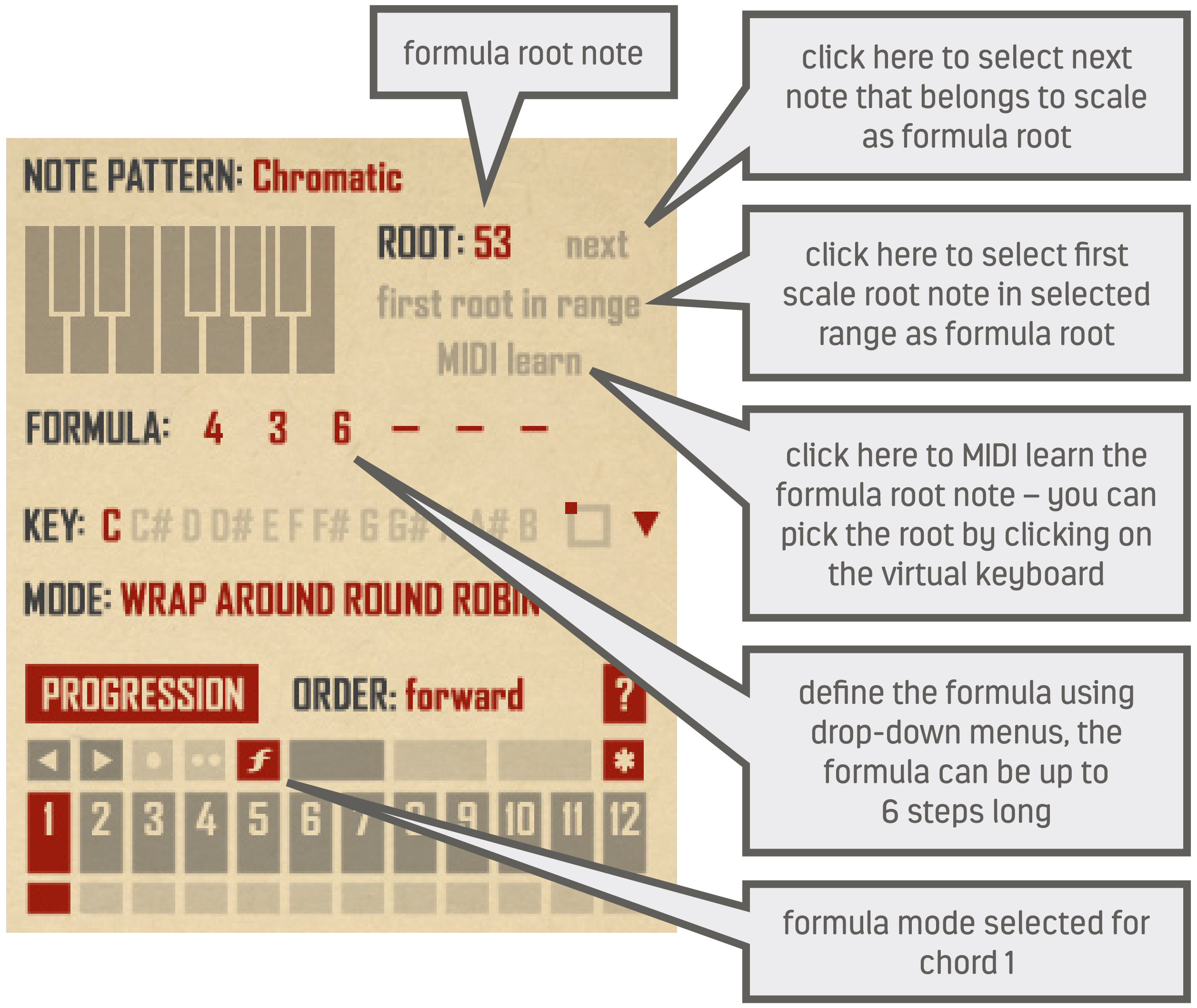
24 tone mask is analogous to 12 tone mask, except the pattern keyboard is extended. Here's a C9th chord programmed in 24 mask:

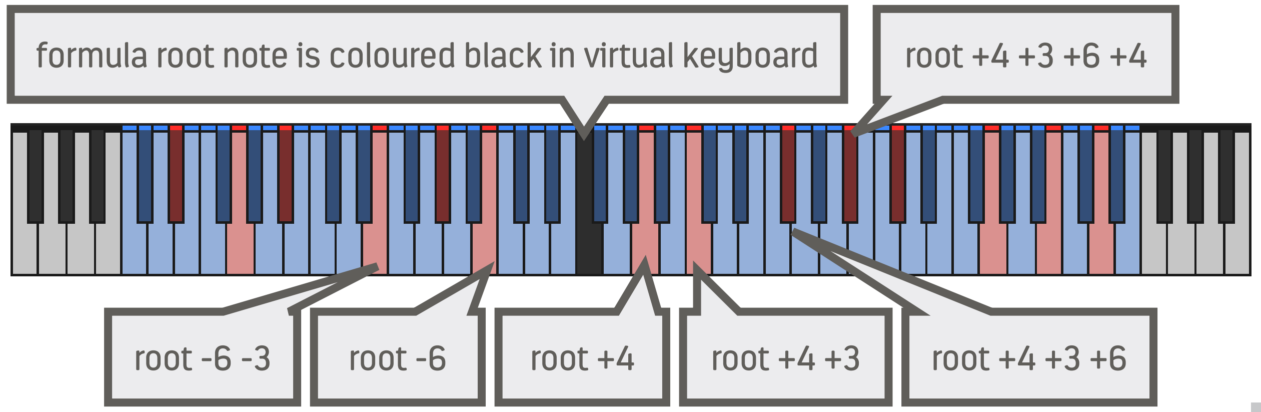

NOTE PATTERN: Ionian (major)
KEY: C C# D D# E F F# G G# A A# B
MODE: WRAP AROUND ROUND ROBIN
PROGRESSION
ORDER: forward
1 2 3 4 5 6 7 8 9 10 11 12

And this is how it is applied to the note range:



In formula mode, pattern keyboard is not used. Instead you can program the sub-pattern formula directly, defining root note and pattern intervals. Unlike with mask modes, the root note can be any note in MIDI range that belongs to selected master pattern / scale. The note pattern is then built around the root note over the whole MIDI range. Formula root number is not the MIDI note number, but the number of note counting only notes that belong to defined scale. Below you can see "54: 4-3-6" formula applied over chromatic scale and how it translates to the note range.


formula root note
click here to select next note that belongs to scale as formula root
click here to select first scale root note in selected range as formula root
click here to MIDI learn the formula root note – you can pick the root by clicking on the virtual keyboard
define the formula using drop-down menus, the formula can be up to 6 steps long
formula mode selected for chord 1
NOTE PATTERN: Chromatic
ROOT: 53
next
first root in range
MIDI learn
FORMULA: 4 3 6 – – –
KEY: C C# D D# E F F# G G# A A# B
MODE: WRAP AROUND ROUND ROBIN
PROGRESSION
ORDER: forward
1 2 3 4 5 6 7 8 9 10 11 12



Formula mode can be useful when generating content to use with microtuning, when you deal with scales larger than 12 tones. It can also give more interesting results with pentatonic or symmetric scales, where mask mode can be too restrictive. It is one of those features that can be entirely ignored if you don't feel you need it.

Programming progression sequence

There are more options in the sequence step menu than just loading one of the defined chords. You can program the machine to create a random chord, or to transform a previously loaded chord. Here's the overview of step program options.

progression pattern 1-12 – load and select (for editing) one of 12 defined chords,

auto pattern – create a pattern/chord using seed derived from cursor or field checksum (in the same way as with sequence ORDER – see page 20), the machine will draw a pseudo-random chord which will be synchronized with field contents (from field) or with current cursors positions (from cursors). The idea is that given cursor configuration or field map will always generate the same chord formula.

create random pattern – generate a random pattern/chord (fitting selected scale). A note here: this function, as well as all randomisation functions in Cracklefield, is using a seed based pseudo-random generator. The seed is being reset according to field checksum every time the transport starts (you press play or record in DAW). So if you use Cracklefield in a project and it starts from the same field contents, all pseudo-random events will be repeated with the same results. In other words, you should get the same sequence every time you render a project.

no change – it's a blank step, previously loaded or generated chord will remain unchanged. It can be used to create a sequence structure when using auto or random patterns. For example, a sequence: random - blank - random - random, translates to: play a random chord for two steps, then play another random chord for one step and a third random chord for one step.

select random step – it's a special function which can introduce some non-linearity to the sequence. When sequencer comes to the 'random step' program, it will jump the sequencer playing position to another randomly picked step (other than 'select random step', in case there are multiple random points) and perform whatever function is assigned to the new step.

mute step – mute the instrument for one progression sequence step. It can be used to introduce pauses to the sequence. All functions will be performed normally, expect for playing the notes which will be skipped.

mute step (second half) or (last quarter) – is a crossover between a blank step and a mute step. It can be used to make short pauses. Mute second half will play the first half of the progression step and mute the second half. Similarly 'last quarter' will only mute the last quarter of the step.

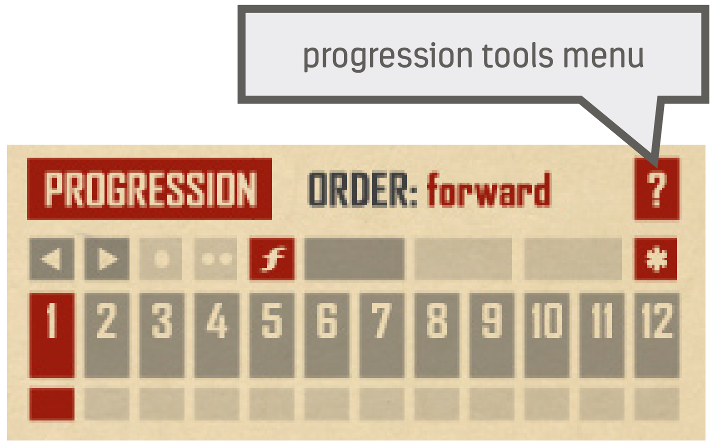
change offset -6 to +6 – change progression chord root note and keep chord structure. For example, following sequence: "chord 1, chord 5, chord 6, chord 4" (assuming we are using default chords where chord root equals chord number) is equivalent to "chord 1, offset +4, offset +1, offset -2". In mask modes offset will wrap around if you push the root note out of the scale length. Offset change can be useful with auto/random patterns to set up a half random - half fixed progression.

auto formula – create random formula-based pattern/chord using cursor configuration as seed. Narrow and wide iterations differ in allowed step range – wide function will allow larger steps, producing stranger patterns. Use to experiment. It will give interesting results with pentatonic or symmetric scales.

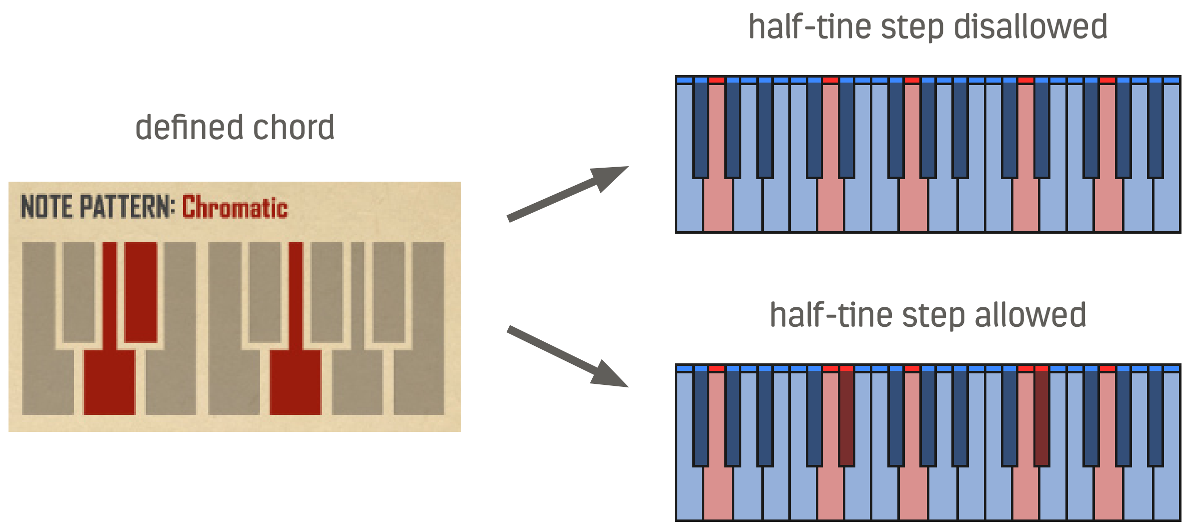
Note that only step options which produce entirely new chord are available for step one of the sequence. This is because functions that only transform a part of chord formula wouldn't make sense without a pre-existing data. The first sequence step pattern is loaded each time you start the instrument or load a preset/snapshot.

Progression tools menu

Click on 'question mark' box to access the drop-down menu with several progression management functions. Here's what is available:



allow/disallow half tone step – sometimes when dealing with random chords and certain scale patterns, the wrapped-around progression pattern may result in two notes in a map that are one semitone away from each other, creating a dissonance. This function removes such dissonant notes from the final pattern. Half tone step is disallowed by default; if you intend to create such a chord, for example, using chromatic scale, change it to 'allow'. Example:

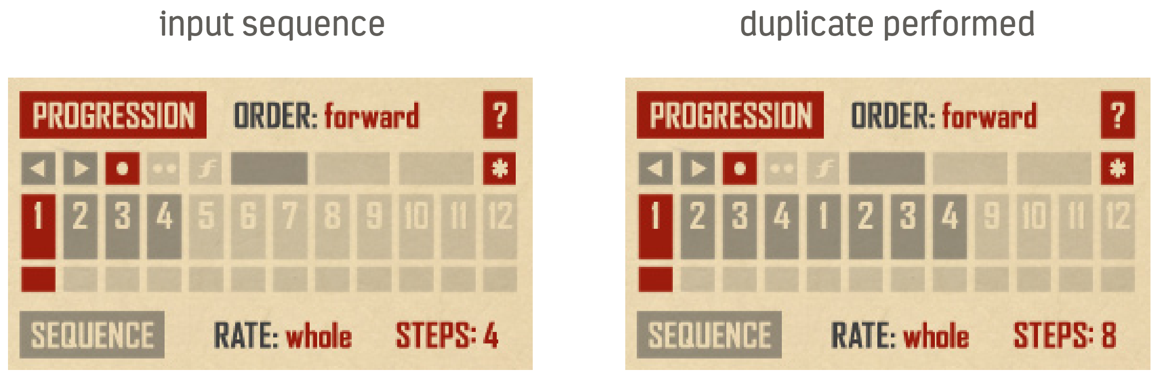


reset progression patterns – erase all 12 patterns and fill them with default formula: root, +2, +2; root +1, +2, +2 and so on. It will erase any manually edited chords, so use it with caution.

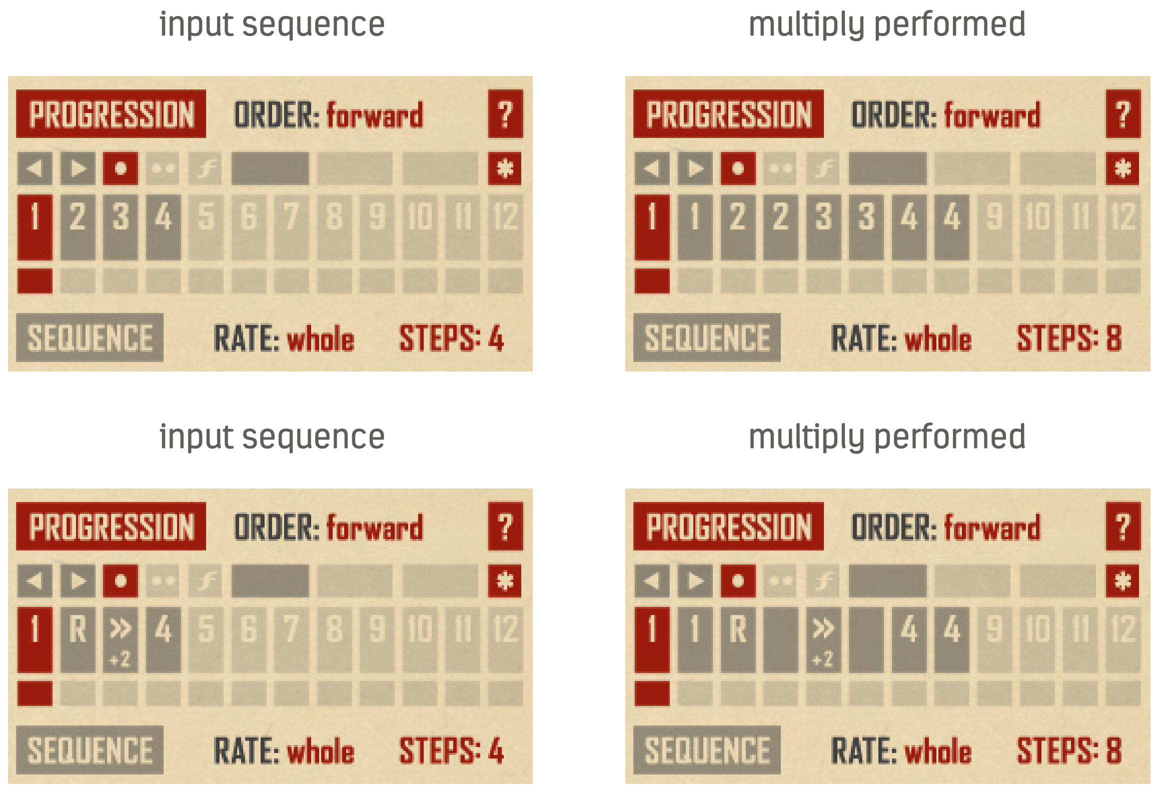
reset progression sequence – overwrites whole progression sequence with default string: chord 1, chord 2, chord 3... and so on. It applies to all 36 steps; defined sequence length is not being changed.

random progression sequence – assigns random chords to all 36 progression sequence steps.

duplicate sequence – repeats the defined sequence, setting sequence length to double (considering sequence is no longer than 18 steps), useful if you want to repeat existing progression but with some minor changes, Example:



multiply sequence – split each step into two (will only work on sequences no longer than 18 steps). Useful to create progressions where there are some changes at half step, like 1-1-5-5-4-6. Steps with random chords, or transforming functions will have blank step as the duplicate. Example:



pick random scale and key – select random scale from set pre-defined pattern, then select random key. It is the same function which is being used by "create random preset". It will prefer minor/major scales (Ionian / Aeolian), also it will be less likely to pick exotic patterns like Pelog.

create random progression – again this function is part of "create random preset". It will erase all user-edited chord patterns replacing them with the default set and then set a random progression. It is much more advanced than "random progression sequence"; it will set up sequence order, length and rate. It can create completely random progressions, including progressions which make use of chord modifying steps, or steps generating random chords. Often it will pick one of the pre-defined common progressions. It will use formula-based chords for pentatonic and symmetric scales; for other patterns it will always use 12 tone mask chords. If you intend to randomise both scale and progression, it would be better to start with the scale, as the progression randomising function can depend on the scale type.

Scales, progressions and tuned notes

Each cursor can have its own tuning in the range of -12 to +12 semitones, which can be set on the SOUND tab using the PITCH controller. When a cursor is in 'use map mode' (being a melodic cursor), its tuning will be added to the note value read from the map, so you can, for example, define a short note range and have some cursors play lowered notes and some cursors play high notes. As such, notes generated by a tuned cursor can go out of the defined note range.

To keep all notes within the defined scale/pattern, every generated note will be checked to see if it fits the pattern. If not, it will be adjusted to the nearest note that belongs to the pattern. When using the progression generator, the master pattern will be restricted to the selected chord, so any generated note will be fit into the currently selected chord mask.

Velocity map

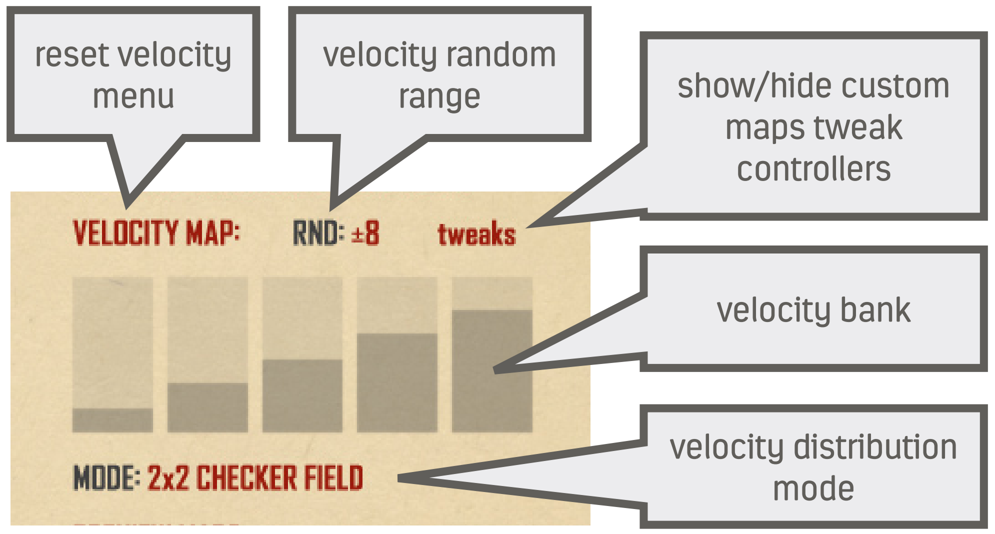
Similarly to the note map, there is velocity map used to determine velocity of generated notes. Unlike note map which is only used when USE NOTE MAP mode is selected, the sequencer always looks up velocity map when generating notes, no matter which mapping mode is selected.

Also ADD parameter is not used, when fetching velocity value from the map, it only affects note numbers.

There are five vertical sliders forming the velocity bank, you can use them to set velocity values used to generate the map. Values will be distributed over the field using one of the distribution modes, as with the note map. Note that setting a velocity to zero will effectively mute some of the cells, which will generate no notes, even when set to filled state.

You can use velocity menu to reset, or randomise velocity bank. Picking "no velocity variation" option will set all sliders to the same value, so each cell will be assigned the same velocity.

Use random velocity range controller to define random velocity variation. If the controller is set to a value other than zero, velocity will be randomised for each generated note. The sequencer will fetch mapped velocity value and change it by random number from defined range.



Available velocity distribution modes:

HORIZONTAL ROUND ROBIN – first map line is filled with first velocity value, next line, next velocity value, when last velocity is reached, it restarts from the beginning.

VERTICAL ROUND ROBIN – first map column is filled with first velocity value, next column, next velocity value, when last velocity is reached, it restarts from the beginning.

HORIZONTAL UP AND DOWN – first map line is filled with first velocity value, next line, next velocity value, when last velocity is reached, it proceeds downward the velocity bank.

VERTICAL UP AND DOWN – first map column is filled with first velocity value, next column, next velocity value, when last velocity is reached, it proceeds downward the velocity bank.

2X2 CHECKER FIELD – the map is divided into 2x2 blocks, following blocks are filled with following velocity values.

3X3 CHECKER FIELD, 4X4 CHECKER FIELD, 5X5 CHECKER FIELD – the same as above for different block sizes.

As with note map, it is possible to manually modify the map. Use CUSTOM and CUSTOM (LEARN) velocity distribution modes, as described with custom note map. Setting both maps (note map and velocity) to CUSTOM (LEARN) lets you record note numbers and velocity numbers at the same time. Note that it is impossible to manually set velocity number to zero.

You can use PREVIEW MAPS menu to see exact velocity assignments over field cells (pick PREVIEW VELOCITY VALUES). You can keep preview enabled, while MIDI learning the map to monitor entered values.

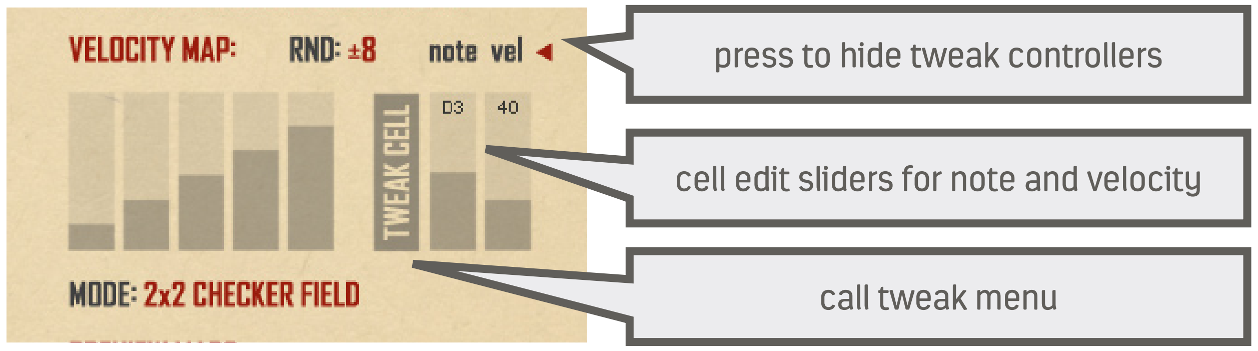
Custom maps

It is possible to manually edit note and/or velocity maps. Following notes can be entered using MIDI keyboard (MIDI learned). Alternatively you can tweak each map cell values manually.

To enter map data using MIDI controller, select CUSTOM (LEARN) mode from one of map distribution menus (note and/or velocity). You can now overwrite map contents by playing notes. Incoming note data will be written to the cell that active cursor is positioned at. After writing a single note, cursor will move forward one step, in its defined direction, wrapping around field edges as a SCANNER. Note that cursor in diagonal mode will not work for map MIDI learn. When you're finished modifying note map, change map distribution mode to CUSTOM. It will preserve entered notes. Changing map mode to any other mode than CUSTOM, will overwrite whole map with automatically generated data. If you happen to overwrite custom map by accident, remember, that you can go back using UNDO function.

Press TWEAKS button to show cell tweak controllers. There are two sliders, which can be used to manually edit map data on a cell selected cursor is at. Using a tweak slider will automatically set corresponding map mode to CUSTOM. You can also use a drop-down menu to process custom maps in different ways, clone map fragments, slide or shuffle cells.

When using custom map, remember that ADD parameter moves map reading point, if you have ADD value assigned to a cursor, it will read from different portion of the map, than it had been recording to. To avoid confusion, set ADD to zero. Also, remember that all notes are being tuned to defined scale before playing, if you record notes that do not belong to the note pattern, they will be aligned. To avoid that, select chromatic scale as the note pattern and disable progression generator.



Mapping modes for percussive sample sets

Percussive sample sets in Cracklefield have different sound variation samples mapped to following notes over the keyboard. They are intended to use in round robin. There are two additional mapping modes to use specially with percussive sequences. The modifier controller for these modes is named TRIM (it is actually the same controller as ADD, just named differently, as it has different function here).

DERIVE NOTE NUMBER FROM CURSOR POSITION

The sequencer looks up the cell index number, to calculate, which note to play, TRIM determines the loop size. If you set TRIM to 2, following cells will tiger the following note sequence: 0,1,0,1,0,10,1... This mapping mode always starts from lowest note. There is REVERSED variation of the mode, which counts notes from the highest note (127) downward. Velocity map is used, as with all mapping modes.

CYCLE NOTES IN ROUND ROBIN

In this mode the sequencer runs a round robin counter for each cursor, treating following notes as following round robin variations. TRIM defines how many round robins to use. Round robin counter is increased each time the cursor moves, so it is increased on empty cells as well. Sequencer will play following notes from 0 upward, until loop limit is reached. Again there is REVERSED variation of the mode, where notes are being played from the highest note downward. Round robin cycle mode is perhaps best suited for percussive sequences.

Note that percussive sound sets in Cracklefield do not have actual 128 round robin variations each. The number of actual round robin varies from 15 to more than 70. However each note has a sound mapped to it, usually by repeating the same set in different order.

Cursor shape tab

In cursor shape tab, there are controllers for shaping volume envelope for sounds assigned to given cursor, as well as cursor and sound timing.

There are three volume envelope controllers: ATTACK, HOLD and TAIL.

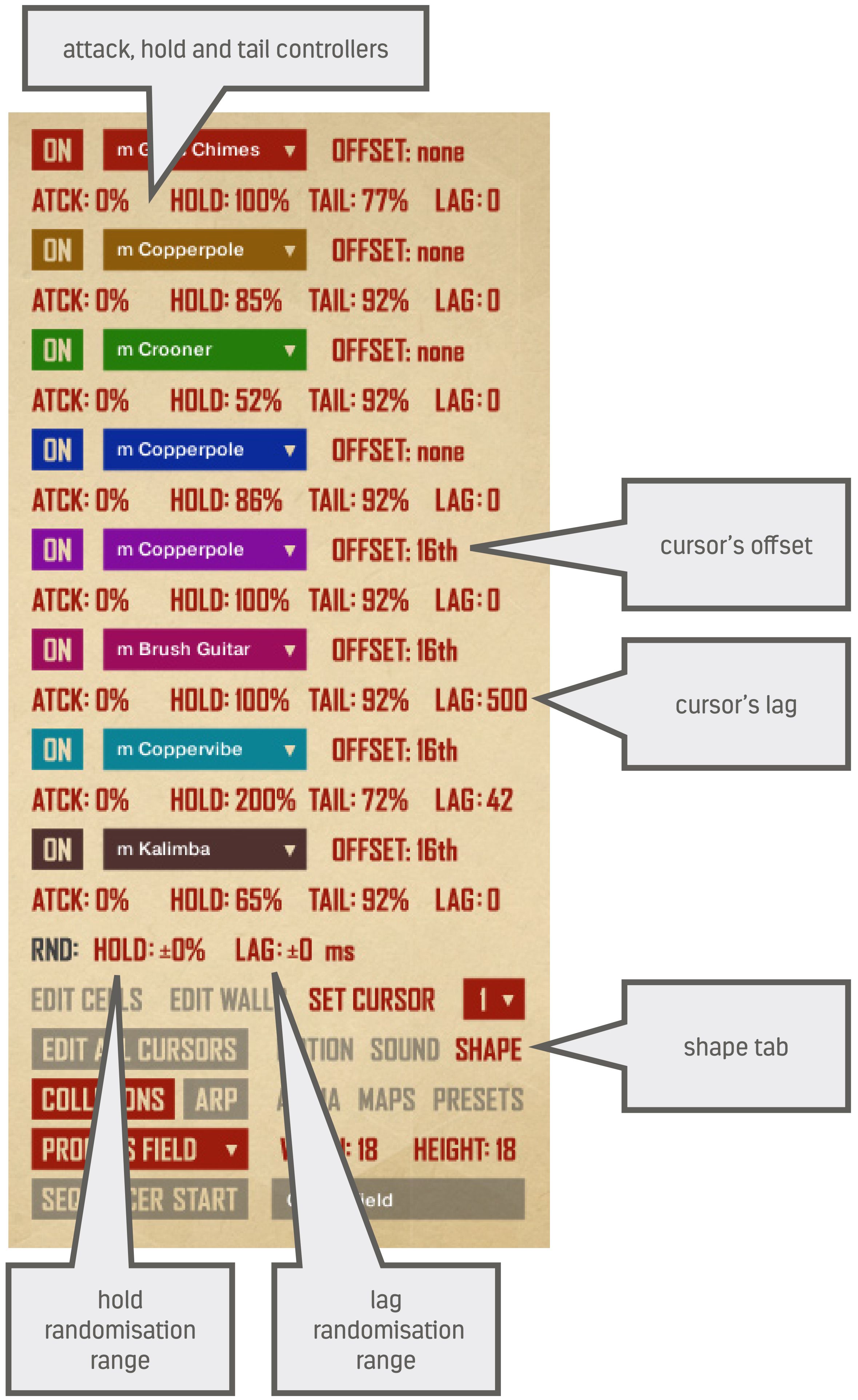
ATTACK and TAIL are assigned to AHDSR envelope's attack and release parameters. These parameters are bound to specific sound set. If you assign the same sound to different cursors, ATTACK and TAIL controller values will be shared by those cursors. If you adjust the parameter for one cursor, it will be changed for all cursors sharing the same sound. The reason behind such design, is to make adding new sounds, as easy as possible. In Cracklefield one sound, equals one group.

HOLD parameter is basically note duration, as percentage of cursor's rate. So, if cursor rate is 16th note, 100% HOLD generates notes with duration of 16th note. This setting is independent to each cursor.

OFFSET parameter has been explained in cursor movement section (page 8).

Finally there's LAG parameter, which defines delay between cursor move time and playing the note. LAG is set in milliseconds and can be used for humanizing or strumming effects. After a cursor moves onto a filled cell, the sequencer calculates note properties and waits for a duration defined by LAG, before playing actual note. Note that in CHORD mode, LAG is applied after each following note in chord, creating strumming effect. Maximum LAG is 500 ms, that is, half a second. LAG value is independent of host's tempo.

There are range randomisation controllers for HOLD and LAG parameters. For each note being played by the sequencer, a random value from defined range will be added to, or subtracted from value defined by given cursor's HOLD and LAG controllers.

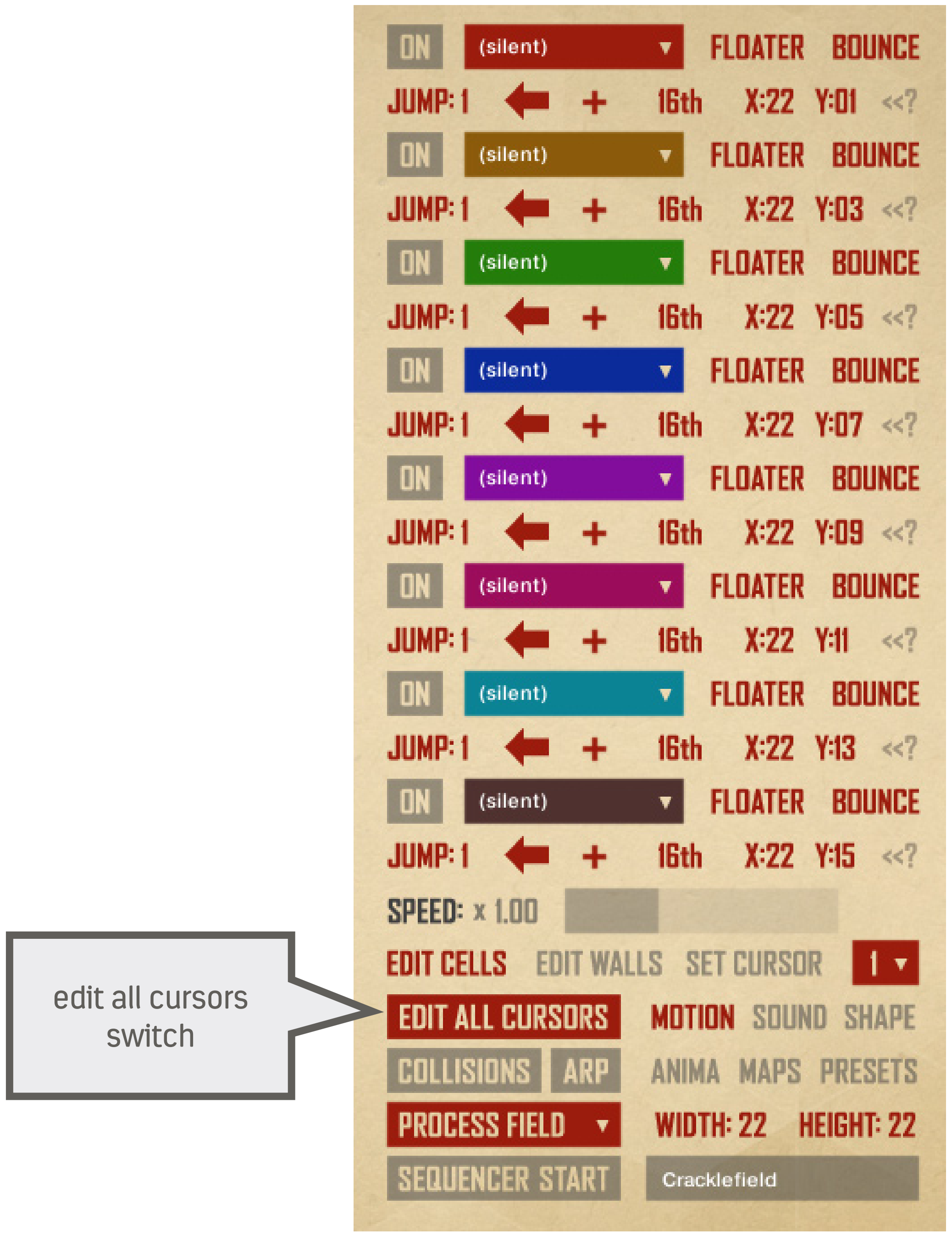


ON m Chimes OFFSET: none
ATCK: 0% HOLD: 100% TAIL: 77% LAG: 0
ON m Copperpole OFFSET: none
ATCK: 0% HOLD: 85% TAIL: 92% LAG: 0
ON m Crooner OFFSET: none
ATCK: 0% HOLD: 52% TAIL: 92% LAG: 0
ON m Copperpole OFFSET: none
ATCK: 0% HOLD: 86% TAIL: 92% LAG: 0
ON m Copperpole OFFSET: 16th
ATCK: 0% HOLD: 100% TAIL: 92% LAG: 0
ON m Brush Guitar OFFSET: 16th
ATCK: 0% HOLD: 100% TAIL: 92% LAG: 500
ON m Coppervibe OFFSET: 16th
ATCK: 0% HOLD: 200% TAIL: 72% LAG: 42
ON m Kalimba OFFSET: 16th
ATCK: 0% HOLD: 65% TAIL: 92% LAG: 0
RND: HOLD: ±0% LAG: ±0 ms
SET CURSOR 1
HEIGHT: 18

Edit all cursors switch

EDIT ALL CURSORS switch can be used, when programming cursor properties. When the switch is on and you edit a cursor parameter, the edited parameter value will be copied to all cursors. This functionality applies to all cursor parameters.

This option can be handy, but it makes it easy to overwrite some settings by accident. If you happen to forget, to turn it off and erase some carefully built setup, use UNDO function from PROCESS FIELD menu.


edit all cursors switch
ON (silent) FLOATER BOUNCE
JUMP: 1 + 16th X:22 Y:01 <<?
ON (silent) FLOATER BOUNCE
JUMP: 1 + 16th X:22 Y:03 <<?
ON (silent) FLOATER BOUNCE
JUMP: 1 + 16th X:22 Y:05 <<?
ON (silent) FLOATER BOUNCE
JUMP: 1 + 16th X:22 Y:07 <<?
ON (silent) FLOATER BOUNCE
JUMP: 1 + 16th X:22 Y:09 <<?
ON (silent) FLOATER BOUNCE
JUMP: 1 + 16th X:22 Y:11 <<?
ON (silent) FLOATER BOUNCE
JUMP: 1 + 16th X:22 Y:13 <<?
ON (silent) FLOATER BOUNCE
JUMP: 1 + 16th X:22 Y:15 <<?
SPEED: x 1.00
EDIT CELLS EDIT WALLS SET CURSOR 1
EDIT ALL CURSORS MOTION SOUND SHAPE
COLLISIONS ARP ANIMA MAPS PRESETS
PROCESS FIELD WIDTH: 22 HEIGHT: 22
SEQUENCER START Cracklefield

Cursor randomisation

On the right side of cursor controllers cluster, there are controllers marked "<<?", visible on MOTION AND SOUND tabs, which bring up cursor randomisation menu. You can choose to apply random set of values to some of cursor properties, or reset all of cursor's controllers. Used with EDIT ALL CURSORS switch on, the function will apply randomisation to all of the cursors, which are turned ON.

<u>Available options:</u>

randomise cursor mapping – pick random cursor mapping option and set random value for ADD/TRIM controller. This function will also reset LAG parameter, using random lag for CHORD mapping mode. Randomiser function will make distinction between melodic and percussive sounds and assign mapping modes accordingly.

randomise cursor motion – pick random values for cursor's direction, diagonal mode and speed (JUMP). JUMP value will be randomised scarcely.

randomise cursor pan – pick random value for cursor panning, when applied to all cursors, the function will try to create a balanced distribution.

randomise cursor pitch – random value for cursor's PITCH controller, when applied to all cursors, value of zero is slightly preferred.

randomise cursor position – place cursor at a random cell. The function will pick a cell within defined field boundaries. If possible, it will also try not to place a cursor at a wall or at another cursor.

randomise cursor rate – pick random value for RATE controller. The function is not very likely to pick triplets and it will not pick 32nd rate. It will reset OFFSET controller, sometimes it will set OFFSET to fraction of cursor's RATE.

randomise cursor sound (sample set) – set random sound. The function makes distinction between melodic and percussive sound and will not substitute one with the other.

randomise cursor type – pick random cursor type and cursor mode (BOUNCE or PASS THROUGH). The function has slight preference for FLOATER type.

randomise cursor volume – pick random value for cursor's VOLUME controller. The function will pick the value from range -12 dB to 0 dB.

randomise cursor arp mod – random value for MOD range in ARP mode.

randomise cursor arp note – assign random arp chord note. Note that, when applied to all cursors, the function will proceed from lowest note upward, assign lowest note to random number of following cursors, assign next note to random number of following cursors and so on. Otherwise the sequence wouldn't be playable in ARP mode.

randomise all cursor settings – do all of the above. This function will not turn cursors ON/OFF and it will not replace a melodic sound with percussive one (balance between melodic and percussive sounds has to be set manually). Cursor sound volume envelope is not randomised either.

reset cursor arp note – reset arp chord note assignment.

reset cursor – reset all cursor parameters to default.

Special options:

These options appear only in randomisation menu for first and fourth cursors. They always apply to set of four following cursors.

symmetrise 4 cursors (twister) – set three following cursor position and motion parameters for a twister symmetric setup, according to the first cursor. This function also copies first cursor's TYPE, MODE, RATE and OFFSET to three following cursors.

symmetrise 4 cursors (mirror) – similar to previous function, but it will mirror cursor position and direction, instead of rotating coordinates.

cue 4 cursors (offset) – set following four cursors RATE and OFFSET parameters, so they move one after another at the same rate. This function only sets rates of quarter or half note, depending on first cursor rate (quarter and smaller will be all rounded to quarter, any longer rate will be rounded to half).

Note that previous functions overwrite OFFSET values, so if you want to do both, pick symmetrise or mirror first.

All three functions ignore EDIT ALL CURSORS switch.

ARP mode

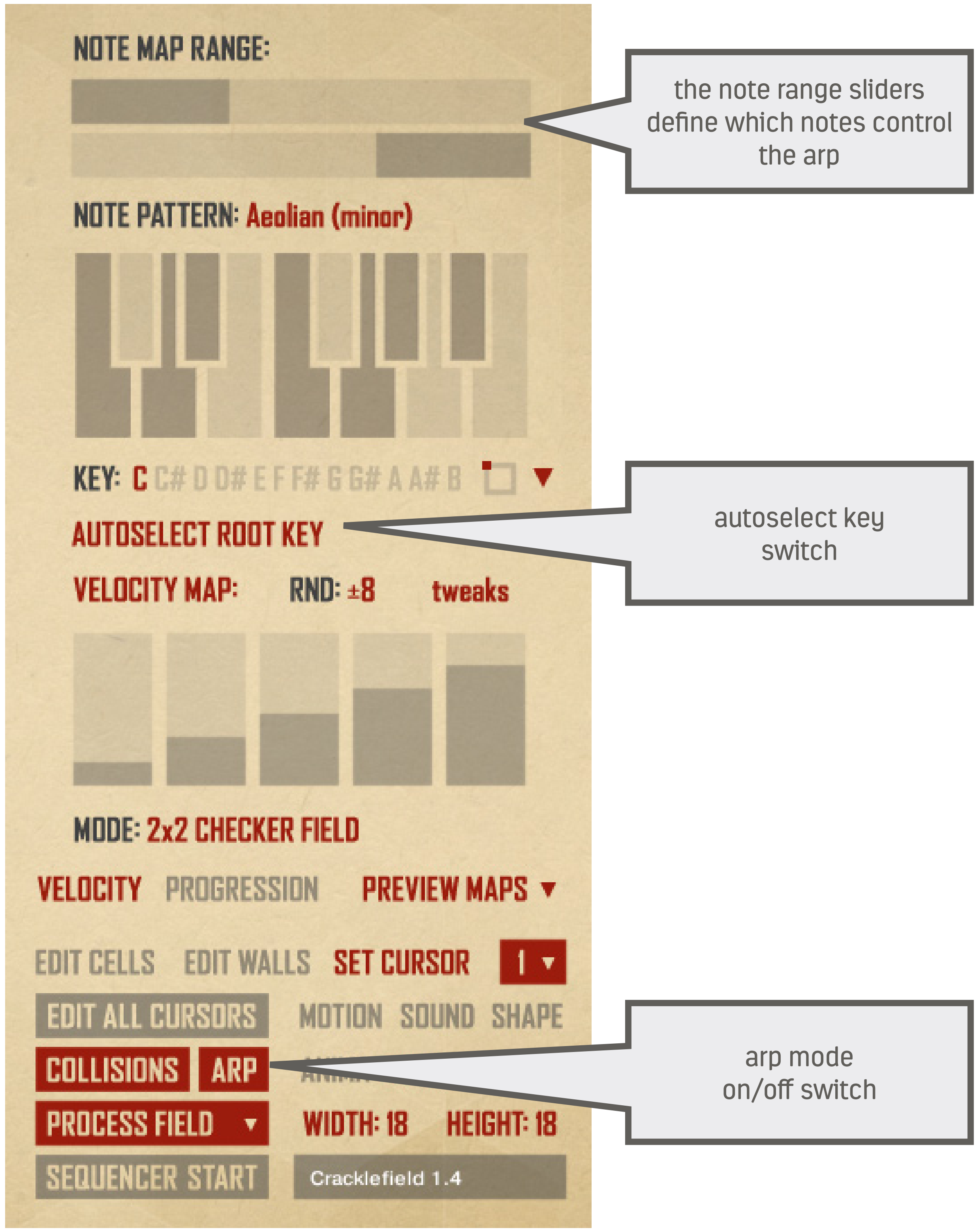
Cracklefield can be set to ARP mode using ARP button. In arp mode cursors do not use mapping modes, but can be assigned to trigger one of the notes from the chord being held on MIDI keyboard. Sequencer START button has no function in arp mode, the sequencer starts when there are notes held in arp area and stops when last note is released.

In ARP mode, note range from NOTE MAP tab is used to define arp range. MIDI notes which belong to defined arp range will control arp engine, notes from outside of the range can be used to play melody by hand. This way the instrument can be configured to control arp with one hand and jam along with it, using the other hand.

Notes generated in arp mode are being adjusted to fit selected note pattern. To disable tuning, select chromatic scale pattern and disable progression generator. Velocity map is used as always.

After selecting arp mode, there will be new option in NOTE MAP tab, AUTOSELECT ROOT KEY. When it’s enabled, the key will be automatically adjusted to match the lowest note of the chord being held in arp range. In arp mode, there are two new controllers in SOUND tab.

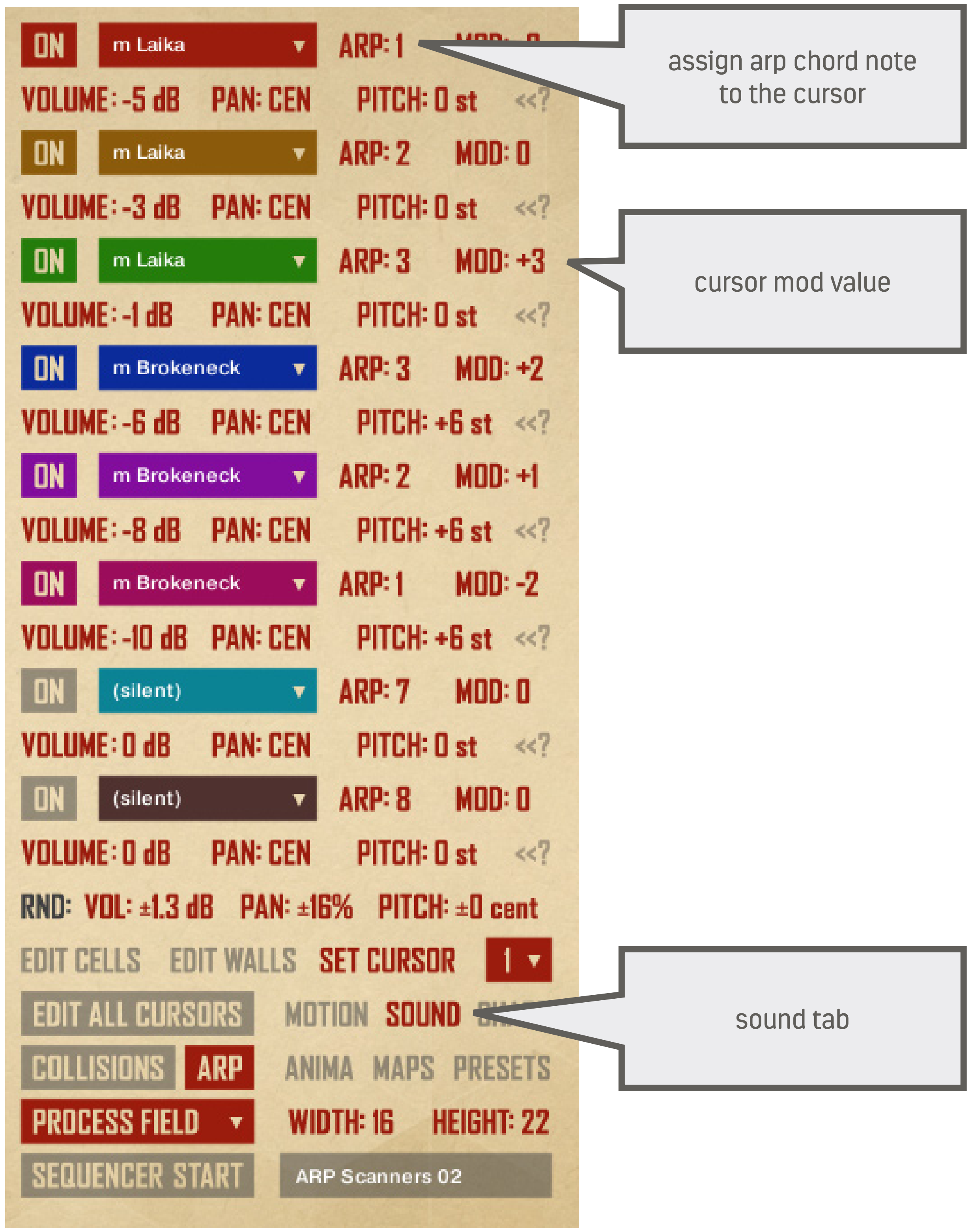
ARP CHORD NOTE is assigned to every cursor, it tells the sequencer which note to play when cursor travels onto a filled cell. ARP 1 is the lowest note being held in defined arp range, ARP 2 is the second lowest note held and so on. You can assign the same arp note to different cursors at the same time. The cursor pitch value will be added to held note number (final note will be aligned to defined scale/note pattern).


NOTE MAP RANGE:
the note range sliders define which notes control the arp
NOTE PATTERN: Aeolian (minor)
KEY: C C# D D# E F F# G G# A A# B
AUTOSELECT ROOT KEY
autoselect key switch
VELOCITY MAP: RND: ±8 tweaks
MODE: 2x2 CHECKER FIELD
VELOCITY PROGRESSION PREVIEW MAPS
EDIT CELLS EDIT WALLS SET CURSOR 1
EDIT ALL CURSORS MOTION SOUND SHAPE
COLLISIONS ARP
arp mode on/off switch
PROCESS FIELD WIDTH: 18 HEIGHT: 18
SEQUENCER START Cracklefield 1.4

ARP MOD controller can be used to set modulation range for each cursor. It works by adding a number from defined range to played note number.

The number is calculated from cursor position, in round robin. So for example, for MOD +3, if cursor position is 1,1, 0 will be added to played note number, for 2,1 position it will be 1, for 3,1 – 2, for 4,1 – 3 and for 5,1 – 0 (it restarts from the beginning).

Together with NOTE PATTERN, ARP MOD can be used to create alternating melody sequences, triggered by just holding one chord.



ON m Laika ▼ ARP: 1
VOLUME: -5 dB PAN: CEN PITCH: 0 st <<?
ON m Laika ▼ ARP: 2 MOD: 0
VOLUME: -3 dB PAN: CEN PITCH: 0 st <<?
ON m Laika ▼ ARP: 3 MOD: +3
VOLUME: -1 dB PAN: CEN PITCH: 0 st <<?
ON m Brokeneck ▼ ARP: 3 MOD: +2
VOLUME: -6 dB PAN: CEN PITCH: +6 st <<?
ON m Brokeneck ▼ ARP: 2 MOD: +1
VOLUME: -8 dB PAN: CEN PITCH: +6 st <<?
ON m Brokeneck ▼ ARP: 1 MOD: -2
VOLUME: -10 dB PAN: CEN PITCH: +6 st <<?
ON (silent) ▼ ARP: 7 MOD: 0
VOLUME: 0 dB PAN: CEN PITCH: 0 st <<?
ON (silent) ▼ ARP: 8 MOD: 0
VOLUME: 0 dB PAN: CEN PITCH: 0 st <<?
RND: VOL: ±1.3 dB PAN: ±16% PITCH: ±0 cent
EDIT CELLS EDIT WALLS SET CURSOR 1 ▼
EDIT ALL CURSORS MOTION SOUND
COLLISIONS ARP ANIMA MAPS PRESETS
PROCESS FIELD ▼ WIDTH: 16 HEIGHT: 22
SEQUENCER START ARP Scanners 02

Field Animators

Field animator is function which can be used to transform entire sequencer field in a single sequence step. You can configure up to four transformations which will be applied at specified rates (in similar way as cursors are being moved). Each animator has two basic parameters: MODE and RATE. MODE specifies what transformation to apply and RATE determines when it is going to be applied. Depending on mode, additional parameters are used: DIRECTION, RULE NUMBER and LIFE RULES. Corresponding controllers will be greyed out when they have no function in selected MODE.

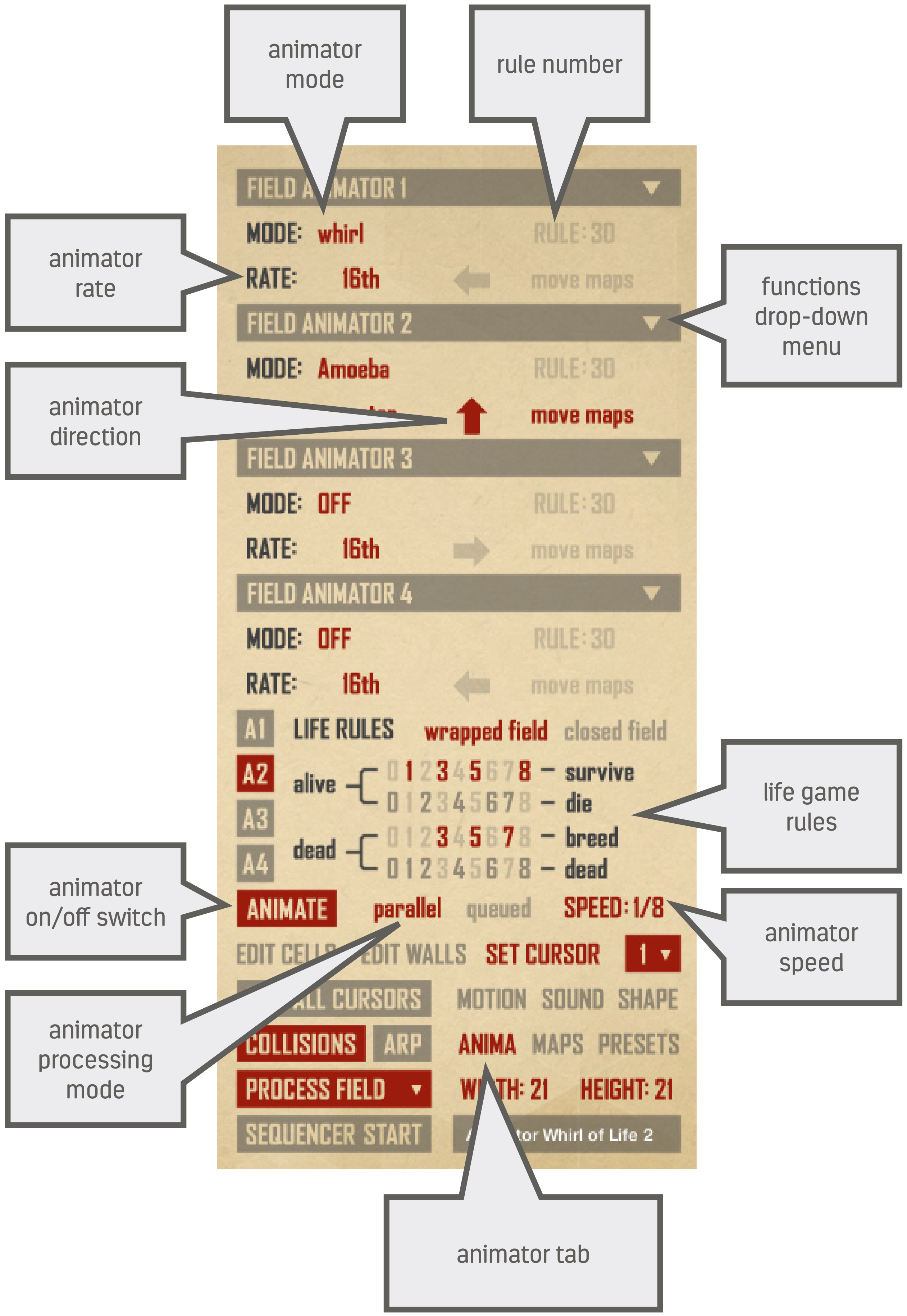
When animator mode is set to OFF, it is disabled. That is, it does nothing. Additionally there is ANIMATE button, which is on/off switch for all animator processing. It can be automated to stop or resume animator processing with MIDI controller. Animator engine has two processing modes: PARALLEL and QUEUED. In parallel mode animators are applied in the same way as cursor movement is sequenced. For example when animator rate is set to 16th, it will be applied each 16th note. When two or more animators are set to the same rate, they will be applied both in one sequencer step.

When cursor movement and animators are scheduled to happen at the same sequencer step, animators are applied first, in numbering order. Animator 1, then Animator 2 and so on. Then cursors are being moved.

In QUEUED mode, as the name suggest, animators are applied in a queue. Only one animator one can applied at one sequencer step and animator RATE specifies time which should pass before applying next animator.

For example, if we have Animator 1 with rate of 16th and Animator 2 with rate of quarter note. In the first sequence step Animator 1 applies, then after a 16th note Animator 2 applies, then after quarter note Animator 1 applies and so on in a loop.

Speed of whole animator sequence can be scaled down with SPEED controller. At animator speed of 1/2, rate of 16th note translates to 8th note, rate of whole note translate to 2x whole note and so on. Animator sequence can be slowed down up to 64 times this way.



FIELD ANIMATOR 1
MODE: whirl RULE: 30
RATE: 16th move maps
FIELD ANIMATOR 2
MODE: Amoeba RULE: 30
move maps
FIELD ANIMATOR 3
MODE: OFF RULE: 30
RATE: 16th move maps
FIELD ANIMATOR 4
MODE: OFF RULE: 30
RATE: 16th move maps
A1 LIFE RULES wrapped field closed field
A2 alive 0 1 2 3 4 5 6 7 8 – survive
0 1 2 3 4 5 6 7 8 – die
A3 dead 0 1 2 3 4 5 6 7 8 – breed
A4 0 1 2 3 4 5 6 7 8 – dead
ANIMATE parallel queued SPEED: 1/8
EDIT CELLS EDIT WALLS SET CURSOR 1
ALL CURSORS MOTION SOUND SHAPE
COLLISIONS ARP ANIMA MAPS PRESETS
PROCESS FIELD WIDTH: 21 HEIGHT: 21
SEQUENCER START tor Whirl of Life 2

Animator modes: MOVE

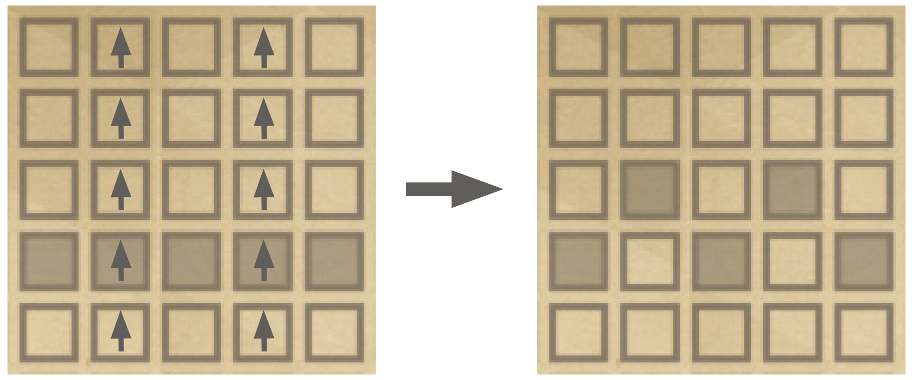
Animator modes can be selected from a drop-down menu. Modes are divided into fore groups labelled: MOVE, GAMES OF LIFE, REBOOT LIFE and ONE DIMENSIONAL GAMES. Modes in MOVE group will re-organize field, cells will be moved around, but overall number of filled cells with remain the same. Modes in GAME groups will run different kinds of cellular automata on the field space, so the field contents will evolve.

With one exception, animators apply to field content only, that is they change cells state, which can be filled or empty. Walls and note maps are left intact. Note maps can be animated using animator sub-mode which will be discussed later.

Animator modes in MOVE group:

scroll field – it is the most basic operation, it slides the whole field one step in specified direction. You can set one of four directions with controller similar to one used with cursors. To achieve diagonal movement, you will need to use two animators. For example, move the field left and then move it up. The field is moved in wrapped around manner. That is, the line/column being moved out of field boundaries, will re-appear at the opposite side of the field.

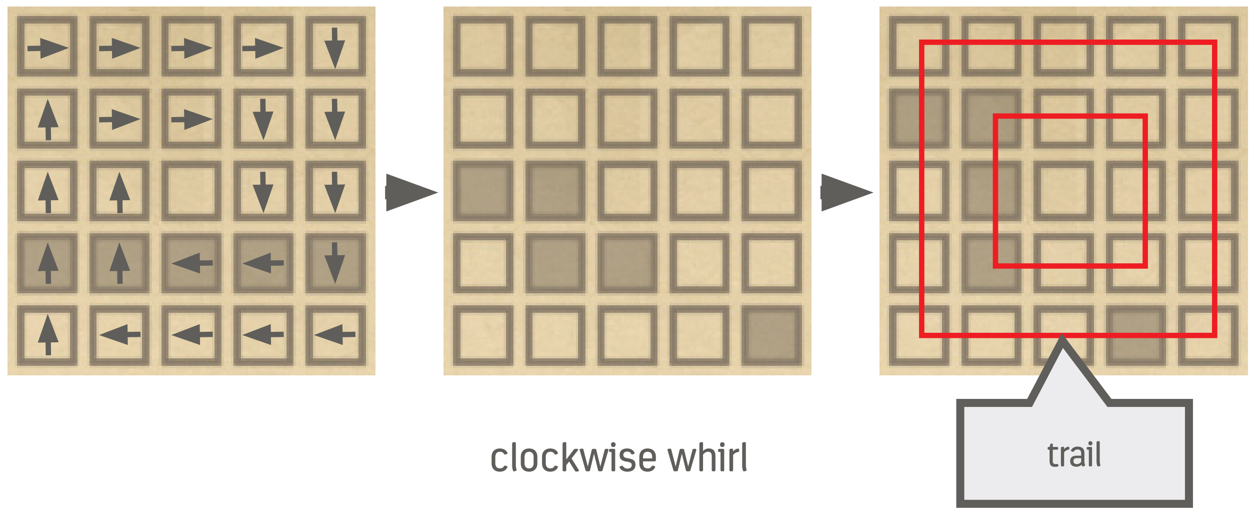
scroll field, even and odd variants – same as above, but only every second row/column is being moved.



scroll field (even), direction = up

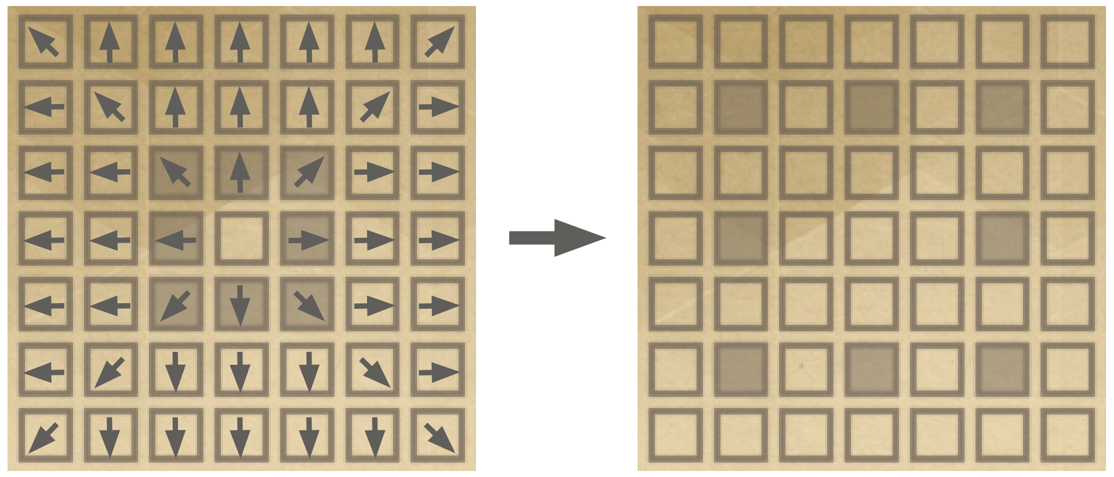
scroll walls – same as in scroll field, but only apply to walls.

whirl – splits the field into rectangular trails and rotates the cells within each trail. This mode has clockwise/counter-clockwise and odd/even variants.



clockwise whirl

wrap-in / wrap-out – splits the field into four parts diagonally, the cells in each part are scrolled toward the field edge (wrap-out) or toward field centre (wrap-in). The field contents wraps around individual lines/columns.



wrap-out

Animator: move maps

Each animator has "move maps" switch. When the switch is activated, a scroll transformation will be applied to note maps only (note map and velocity map – see page 15). It can be handy to introduce note variation for a setup with beacon cursors and moving field.

Move maps is a separate, parallel function, when animator mode is set to anything else than OFF, it does scroll note maps in animator direction. If, for example, animator is set to whirl with "move maps" enabled, it will apply whirl to field contents (cell on/off state) and apply scroll to the note maps.

If you want given animator to only scroll the maps, use "null" animator mode (last one in MOVE group). It does nothing to field contents and is there just for this purpose.

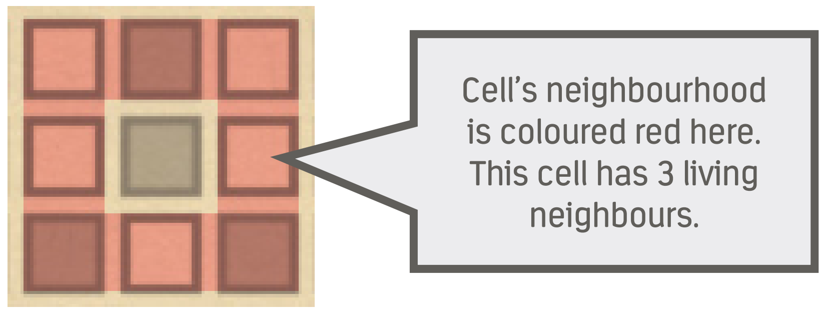
If you enable preview maps in MAPS tab, you can observe how note values are being moved across the field.

Note that when using progression generator, note map will be overwritten at every progression sequence step, as progression is applied after animator, the effect of moving note map will be lost at given step. However it does not apply to velocity map.

Animator modes: GAMES OF LIFE

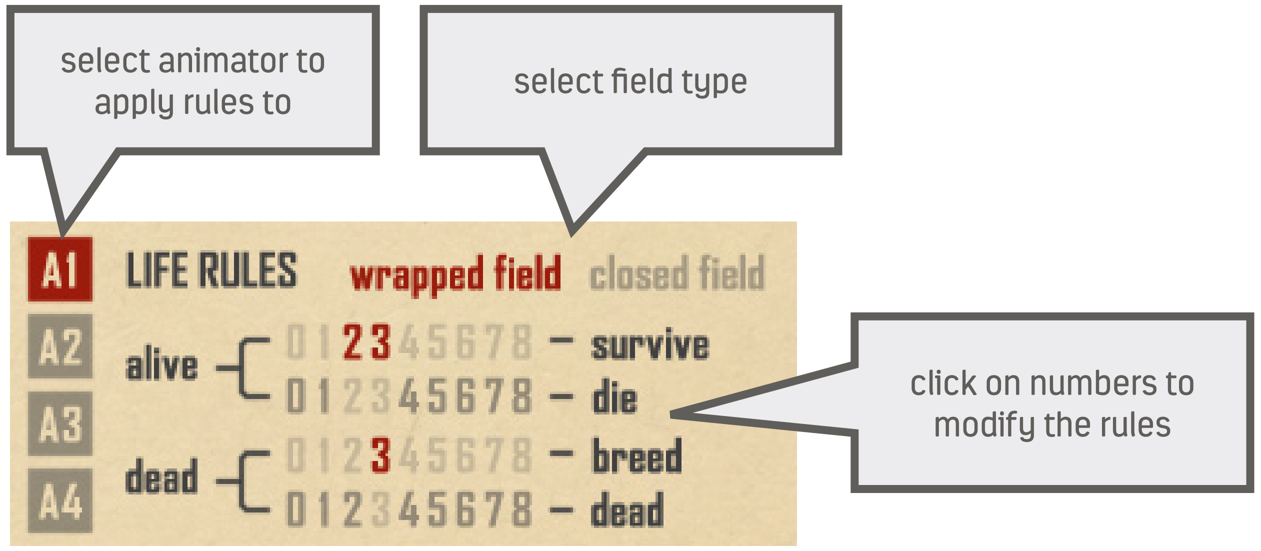
"Game of Life" invented by John Conway is, no doubt, the best known cellular automaton. It is based on two-dimensional grid, where state of a cell in next generation step depends on how many neighbour cells are filled. In Game of Life convention, a filled cell is named "alive" and an empty cell is named "dead". When a live cell has more than 3 live neighbours it dies because of overpopulation. When a live cell has less than 2 live neighbours it dies of underpopulation. When a dead cell has exactly 3 live neighbours, it becomes alive by reproduction.

The game rules can be denoted as "23/3" – live cell survives when it has 2 or 3 neighbours, a new cell is born when it has 3 neighbours. A cell can have up to 8 neighbours, so there are many possible variants of the game, as you can define different conditions of birth and survival. For example 23/36, known as High Life, is game where a live cell survives when it has 2 or 3 neighbours and a new cell is born when it has 3 or 6 neighbours.


Cell's neighbourhood is coloured red here. This cell has 3 living neighbours.

In Cracklefield you can program any game of life variant. Both survival and birth conditions can address up to 9 neighbourhood values (0 to 8 neighbours), which gives us over 250.000 possible combinations. Actually the classic set of rules is not very interesting for field sequencing, as many initial patterns result in all cells dying out quickly.

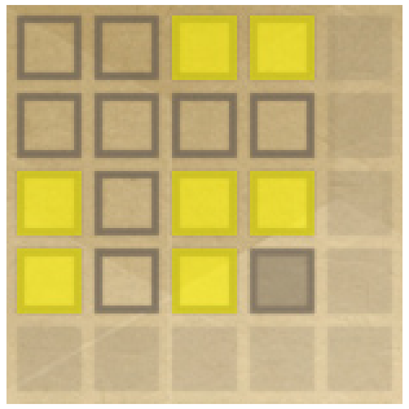
In GAMES OF LIFE modes group there are several preprogrammed rules, which include the most known ones (classic one is named "Conway's Life") and several rules, which I though can be useful for the context. Alternatively you can select "custom life game" and configure rules by hand using rule programmer. Also you can select ready-made rule and just modify it using the programmer, animator mode will be set to "custom life game" automatically once you use rule programmer.

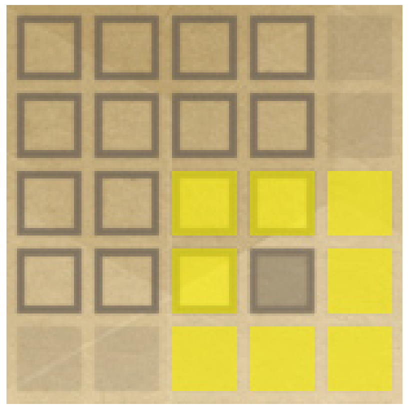


game of life rule programmer – classic 23/3 rule is set here

To set game of life rules, first select a pre-programmed life animator mode or "custom life game" mode. Then select animator number in rule programmer. If a not-applicable mode is selected, rule programmer controllers will be greyed out. Now click on rule numbers to set them. The actual controller type used here is X-Y pad, so you can hold mouse button down and drag the cursor around as you would in programming a sequence table in Kontakt.

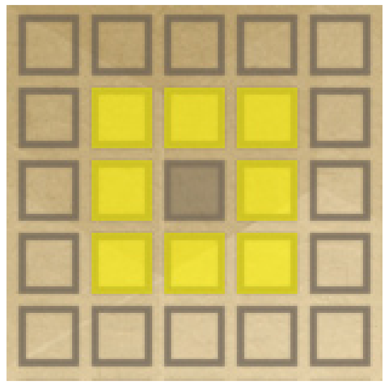
Also there's field type switch, where you can specify how to count number of neighbours for a cell located at the edge of the field. In wrapped mode, cells at the opposite side of the field are considered neighbours. In closed field mode, neighbour cells expected outside of field boundaries, are considered dead.

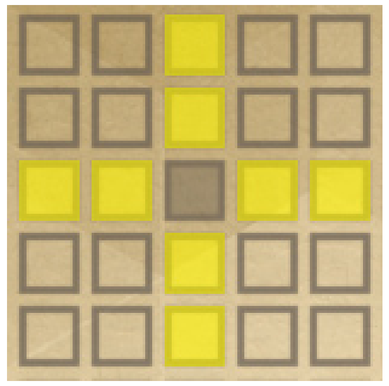

wrapped field neighbourhood

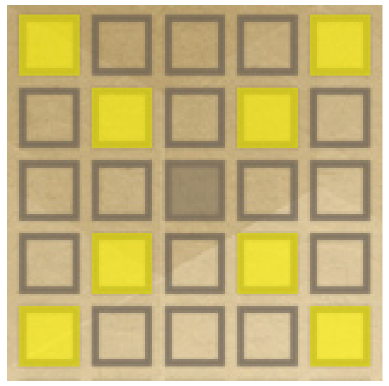

closed field neighbourhood

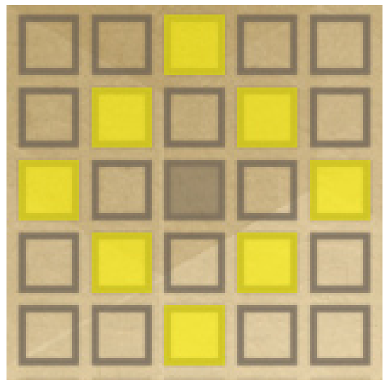
So a cell located on field corner can only have up to 3 neighbours in closed mode.

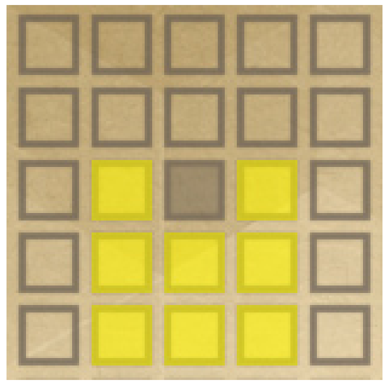
Additionally, for the sake of experimentation, there are four neighbourhood shape variants of the custom life mode. They are named t-neighbourhood, x-neighbourhood, o-neighbourhood and u-neighbourhood. They differ in which cells are considered neighbours, as illustrated below.


regular neighbourhood


t-neighbourhood


x-neighbourhood


o-neighbourhood


u-neighbourhood

As you can see u-neighbourhood is non-symmetric/directional. It is using direction controller to set one of four headings.

Considering that you can combine different life game rules by queuing animators, exploration possibilities are nearly endless.

Animator modes: REBOOT LIFE

Sometimes game of life modes seem to make the field evolve endlessly or create a loop, but often all cells die eventually. This is not quite good for the sake of musical sequencer, as it may just go silent at some point. As counter-measure for such situation, I added REBOOT group of animator modes. A reboot mode will examine the field contents and if there are no live cells at all, it will populate an initial pattern.

Available modes:

restart – at the beginning of the sequence, field contents is being saved. When the field becomes empty, this animator mode will restore the initial pattern.

breed x cells – create random block consisting of specified number of alive cells. The cells will be grouped together.

breed random block – create random block consisting of random number of cells (1–16).

Animator modes: one dimensional games

Finally there's group of one dimensional games. Here the cell state depends on state of three neighbour cells in previous generation. The automaton mechanics has been described on pages 5–6. While thus far it has been used to create a static field, you can now use this kind of automaton to make a scrolling field.

This is a directional animator mode, direction controller can be used to specify which way to unroll the rule. For example, if we set direction to the left, the column at the right edge of the field will be used as parent seed. In each step the whole field will be scrolled one step to the left, erasing the column at the left edge of the field. The new column at the right edge will evolve from parent column preceding it.

You can select a rule number from the list in the drop down menu, which includes most known and interesting ones. Also you can use rule number controller to set any of 256 possible rules.

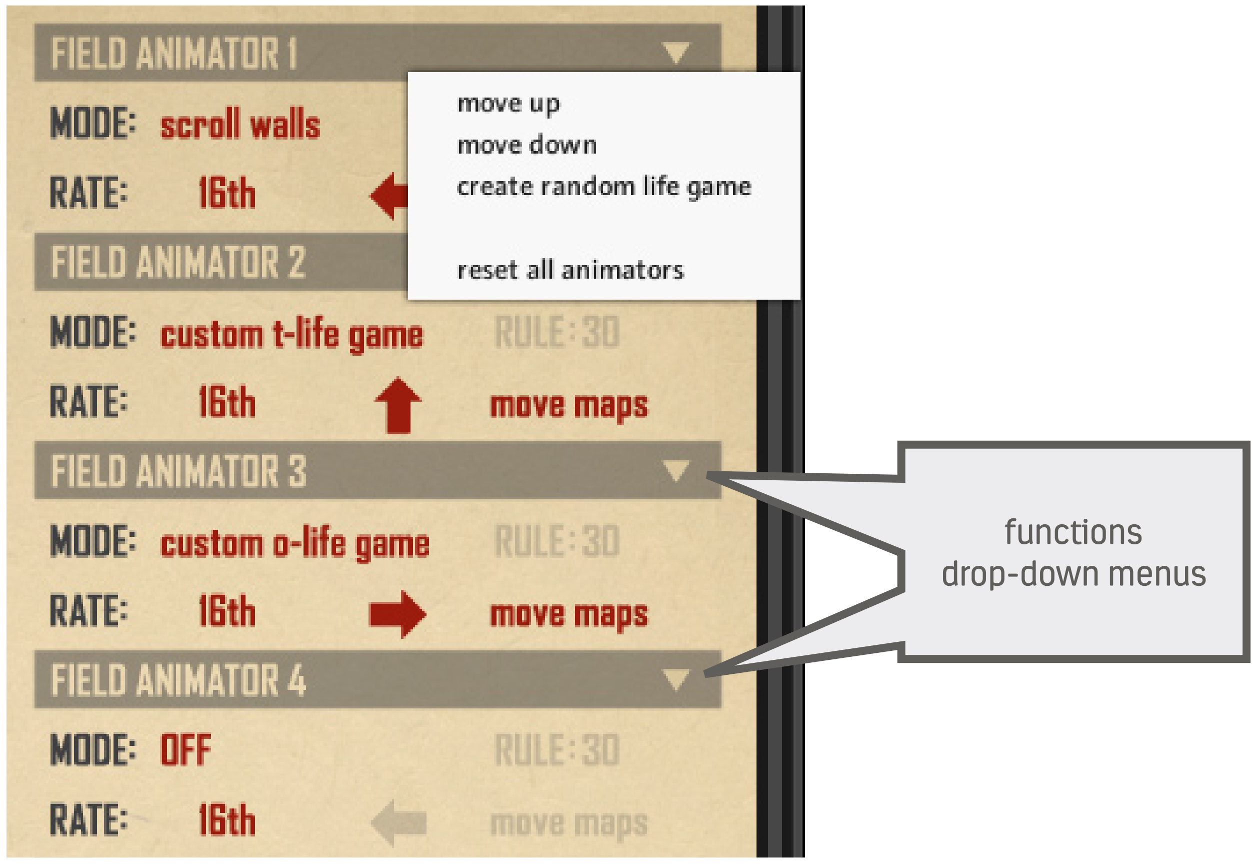
Animator functions

On the right side of each animator label there's functions drop-down menu. There are several automated tasks you can use for animator configuration:

move up / move down – in many cases the order of animators has impact on pattern evolution, especially when using games. These functions let you re-order animators quickly.

create random life game – selects life game mode and creates a random set of rules.

reset all animators – clears all animator configuration. Use with caution, there's no undo.



The sounds of Cracklefield

Cracklefield comes with a selection of experimental, unique or unusual sounds. All samples are in WAV format, 44.1 kHz, 24 bit. They take 940MB of drive space.

Sample sets are divided to melodic and percussive, a prefix "m" has been used to mark melodic sounds and percussive sounds have prefix "p".

List of melodic sounds:

Brokeneck – an old acoustic guitar with broken neck. It had been repaired to some extent, but the instrument became somewhat unstable. The sound box has been filled with plastic foam strips to reduce resonance. It has been recorded at very close range.

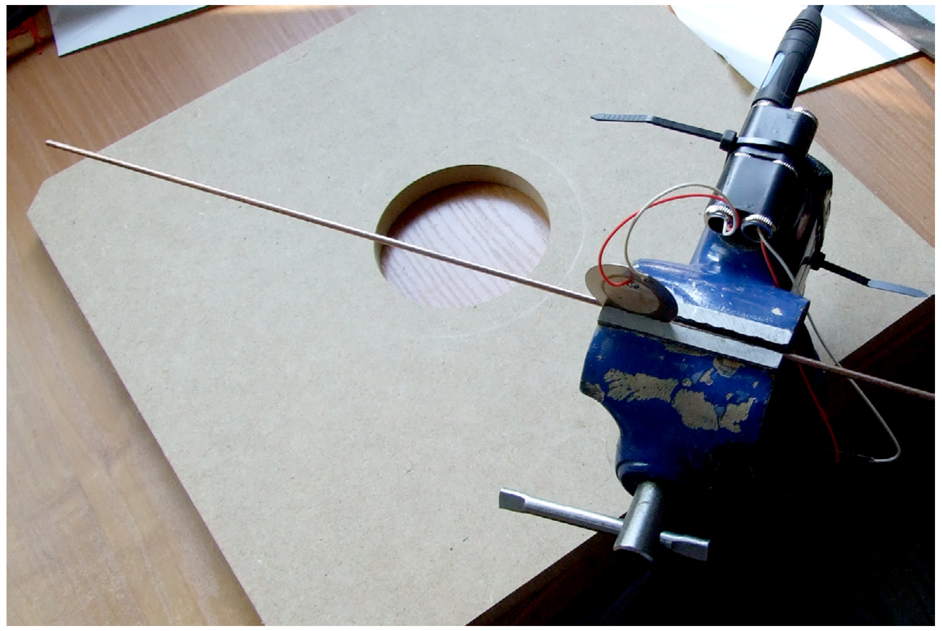
Brush Guitar – an electric guitar, which had the strings damped by a large paintbrush positioned by the bridge. It has very short decay and spiky, funky sound.

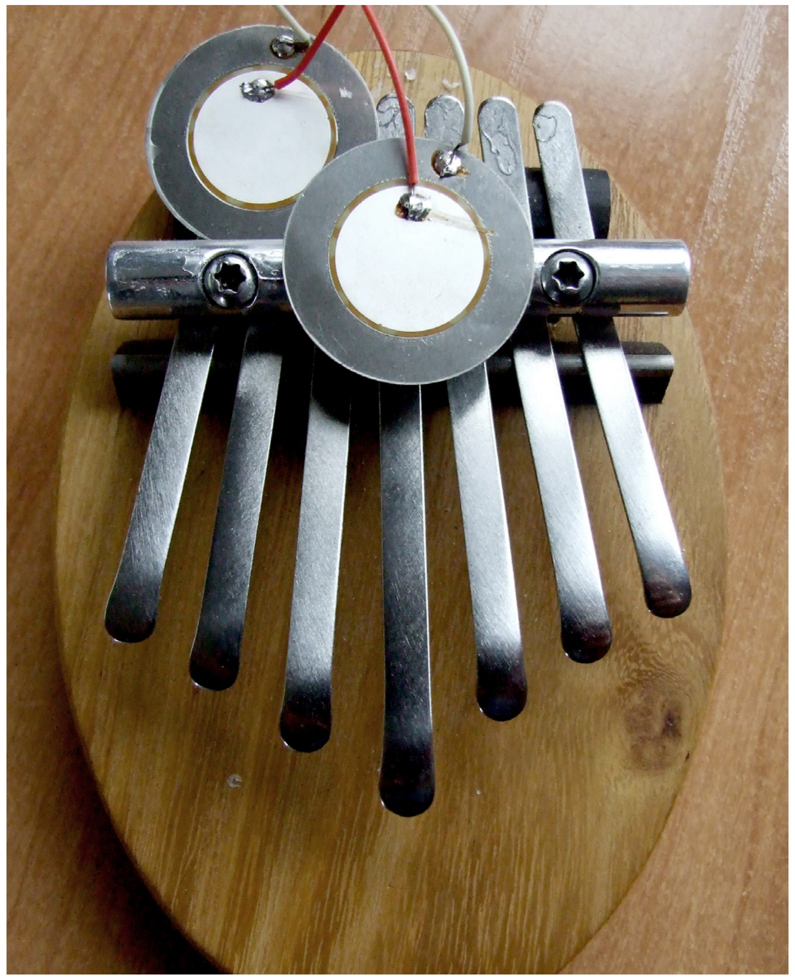
Copperpole and Coppervibe – mallet type sound, generated by a copper rod. The rod has been fastened in a table top bench vice together with a contact microphone. Played with fingers, it generated long lasting tones, mainly infrasound, the time until the full stop was more than 1 minute for each tone. Several clean tones have been captured, they have been then digitally up-tuned and edited to form a sample set. Coppervibe set is built of the same samples but has a tempo synchronized vibrato and has been equalized for a darker tone.

Crooner – is the only fully electronic set of the lot. It's the sound of the singing filter. Tones from self-resonating Moog LP filter (Moogerfooger) have been captured and edited digitally. The volume envelope has been added in audio editor.

Glass Chimes – recorded from a large, perfectly tubular, glass vase. Partly filled with water to tune it. It has been played with a wooden stick.

Kalimba – 7 tone kalimba, made in Poland. Recorded through contact microphones, which have been glued to the body of the instrument, giving it deep and detailed sound.


copperpole


kalimba

List of melodic sounds continued:

Laika – an old soviet balalaika, made in Leningrad. The box was broken, giving the instrument intriguing, hollow sound. The original very old and rusty strings have been used, as well as a set of new Rotosound ones.

Paperfuzz – an electric guitar. Two strips of paper have been slipped in between strings at the bridge, creating a kind of mechanical fuzz.

Renchbass – Jackson Kelly electric bass, placed horizontally and played as a zither with a set of chromium wrenches.

Szczotron – mallet-like sound derived from vocal samples (yes, it was my voice, that has been used), shaped and tuned digitally.

Tesla Caster – electric guitar, customized Telecaster clone, with Tesla pickups. Played gently with fingers in a wool glove, for that fuzzy warm tone.

List of percussive sounds:

Buka – a darbuka, played loosely with fingers.

Crash Fingers – crash cymbal, played with fingers.

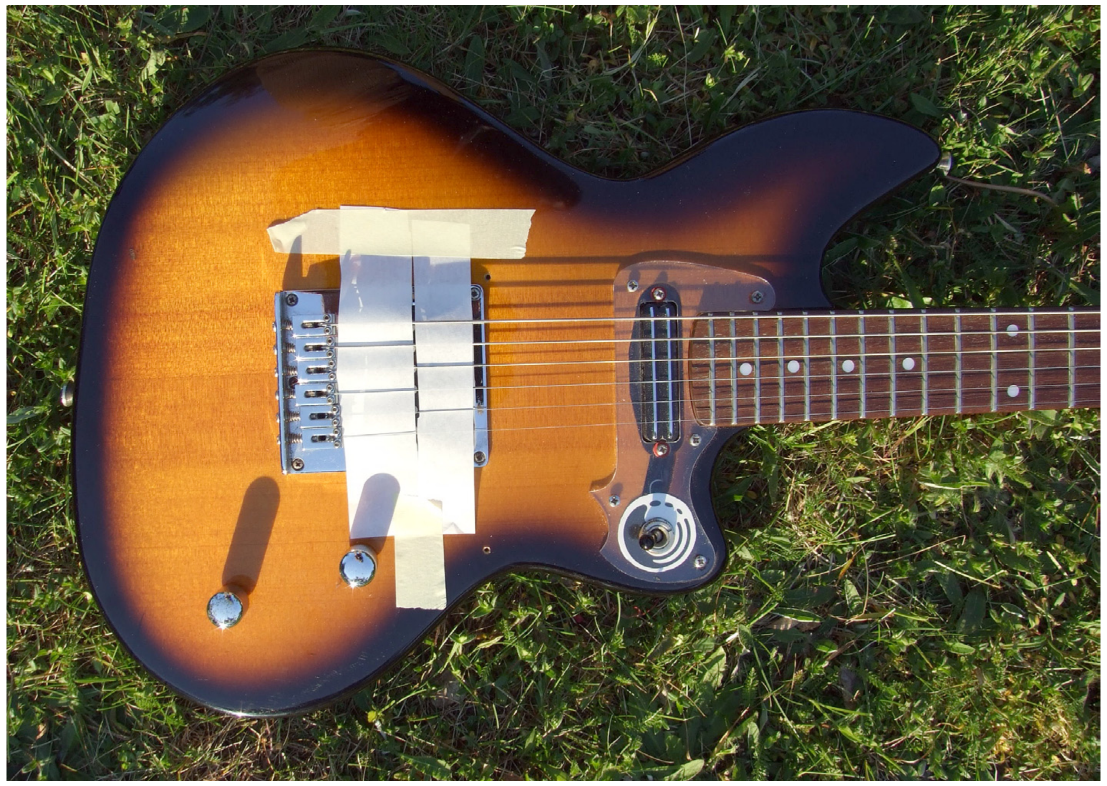
Crash Mellow – crash cymbal, fuzzy splashes played by a variety of unusual objects.

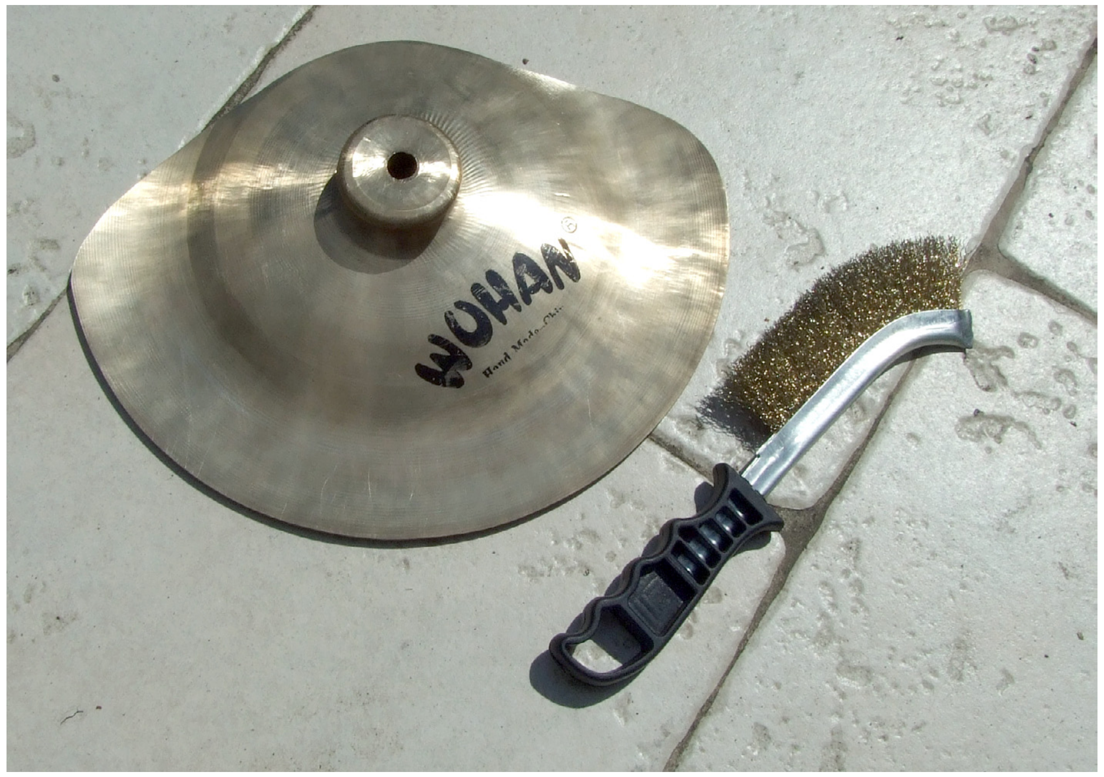
Crash Rider – crash cymbal, played with variety of unusual sticks, recorded at very close range.

Crash Softly – crash cymbal, played gently and quietly with a very light, stick-shaped piece of wood.

Half-China – china splash cymbal, with a large part of body cut off. Played with a metal wire brush.

Medicine-Man – large african ethnic drum, with surprisingly dull sound. Made of very light wood, the two membranes were made from some animal skin, which had quite a few hairy spots. Smashed violently with a fat wooden stick.


paperfuzz



half-china

List of percussive sounds (continued):

Pillow Punch – yes, it's a sleeping pillow being punched with fists. Played back at lower samplerate.

Riceshakes – a shaker. Plastic container filled with rice.

Silverball – exercise rubber-plastic ball. Smashed with thick wooden ruler.

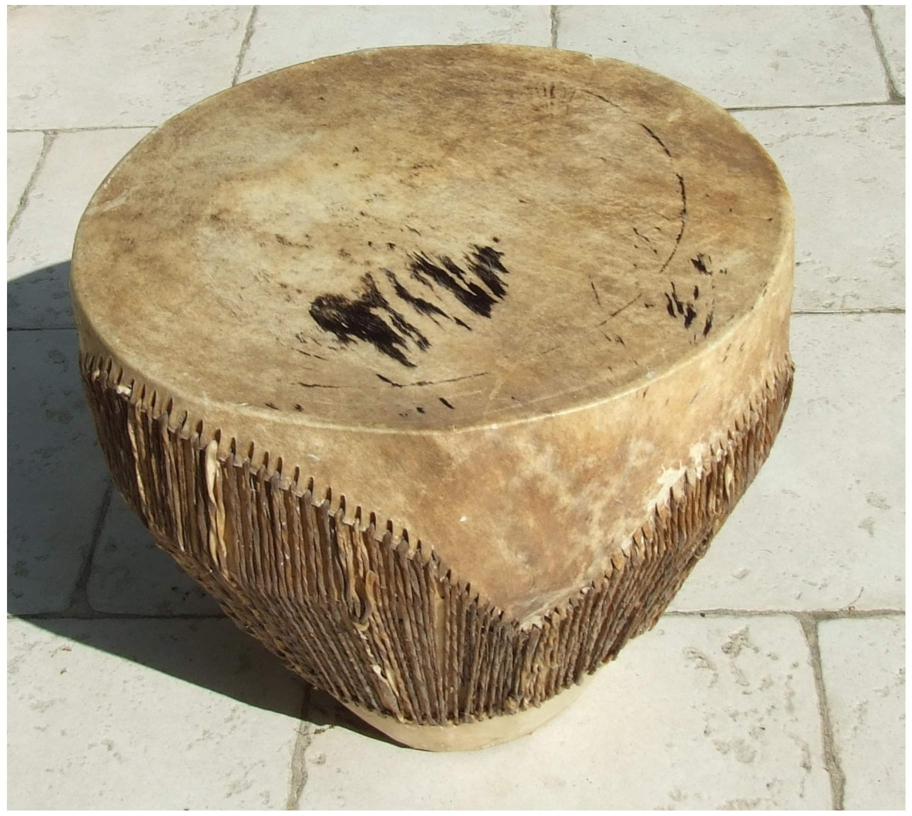
Teashakes – a shaker. Metal container filled with dry tea leaves.

Tom Fingers – medium floor tom. Played gently with fingers.

Tom Muted – medium floor tom. Muted by putting swirls of rope around the membrane. Played with sticks.

Tubelets – short plastic tubes. The opening of the tube was being hit with open hand, making a kind of pop sound. Two sizes were used for high and low variation.

silverball


medicine-man

the tom

Adding user sound sets

Cracklefield has been designed, to make it relatively easy to expand the sample pool. Each sound-set takes exactly one group, new sounds can be added by creating new groups and adding sample maps.

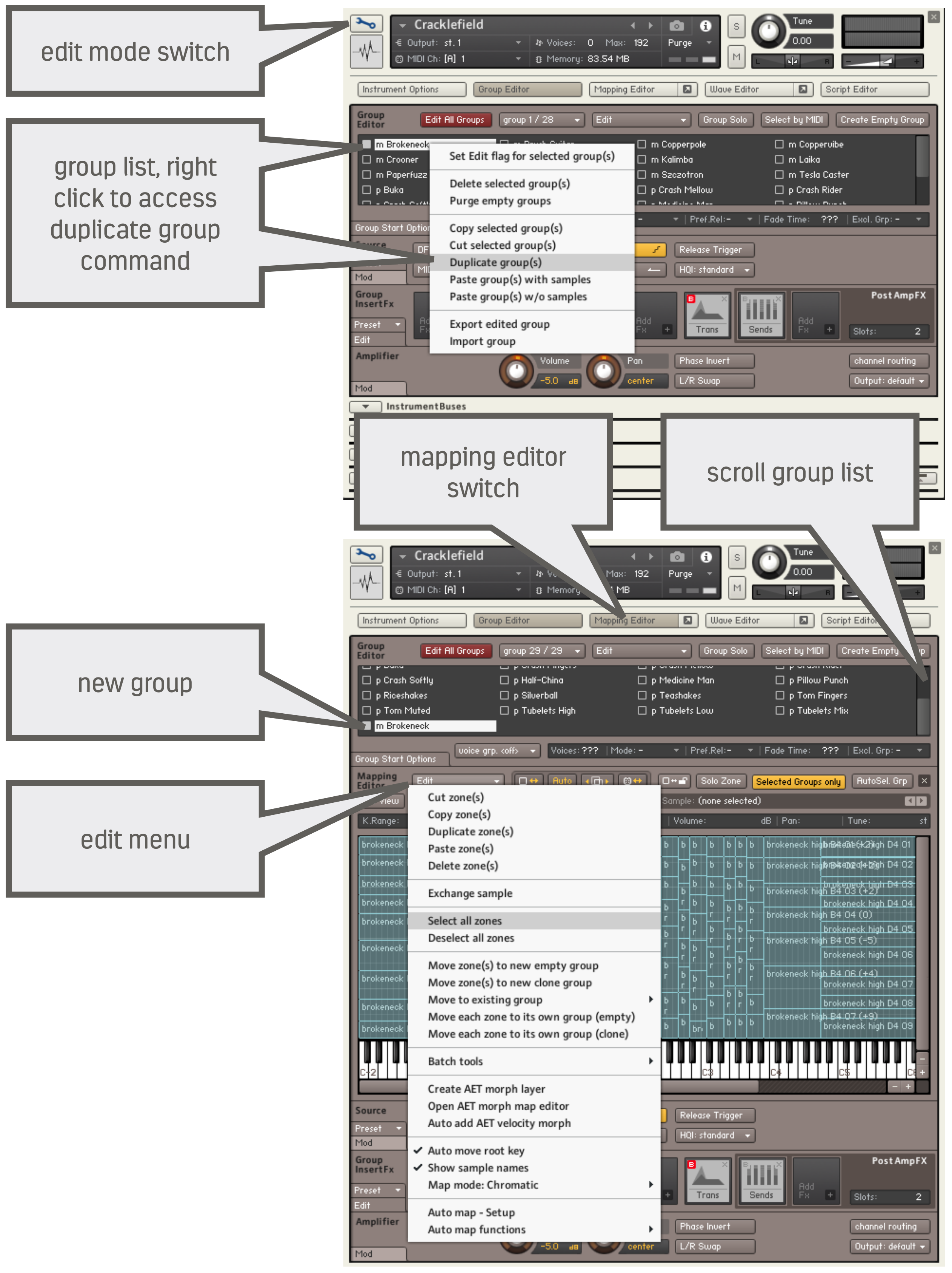
To add a sound, first switch to edit mode, by pressing “the wrench” button. You will see the list of groups. To preserve the effects and modulation, create new group by duplicating existing one. Right click on existing group and pick “duplicate group”. Duplicate a melodic group, if you want to add a melodic sound (note that percussive groups have “tracking” switch set off). Scroll down the group list, newly created group will be at the bottom, it will have the same name as group it has been cloned from. Click on new group to select it, click on mapping editor, to show the sample maps. Pick “select all zones” from “edit” menu, then pick “delete zones” from the same menu.

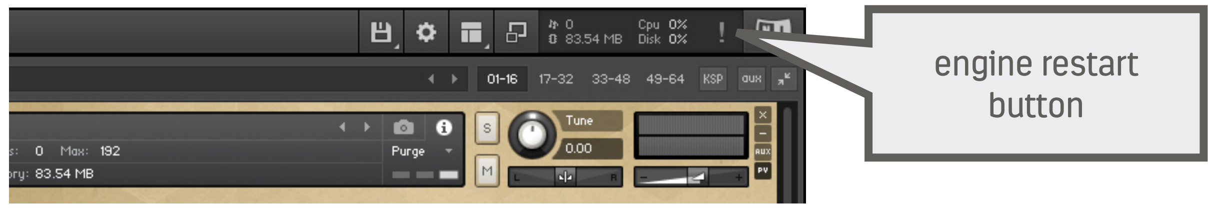
Now you have an empty group. You can double click the group name to re-name it. Sound set names displayed in the Cracklefield sound menu are the group names – you can change sound names by renaming groups. New sounds can be mapped by hand or copied from another instrument (edit menu in mapping editor › select all zones › copy zones, then use paste zones in Cracklefield).

To make added sound appear in sound menu, save and re-load the instrument, or press engine restart button (exclamation mark).

Note: Cracklefield makes distinction between melodic and percussive sounds. Percussive sounds are recognized by examining first group effect slot, if there is “Inverter” loaded in the first slot, the group is tagged as percussive. It is used by randomiser function, so it won’t substitute melodic sound with percussive one and by MIDI recorder to convert note number based round robins to a sequence of the same note number for percussive sounds.

It may be a good idea, not to overwrite the original Cracklefield.nki patch. Just in case, there is the copy in ‘backup’ folder. If you need to restore the original patch, copy the backup file to the root instrument folder.





Preset system

Cracklefield can save/load presets in its own format. Presets can be stored externally in NKA files or internally in one of 12 memory slots, available via key-switches. Of course, one can always use Kontakt snapshot system. Advantage of presets over snapshots, is that presets can be loaded without stopping the sequencer, also you can choose which parts of data to load, you can, for example, load only the field map, switching field contents by key-switches, while the sequencer is running. Disadvantage of a preset is, that it doesn't store effects configuration.

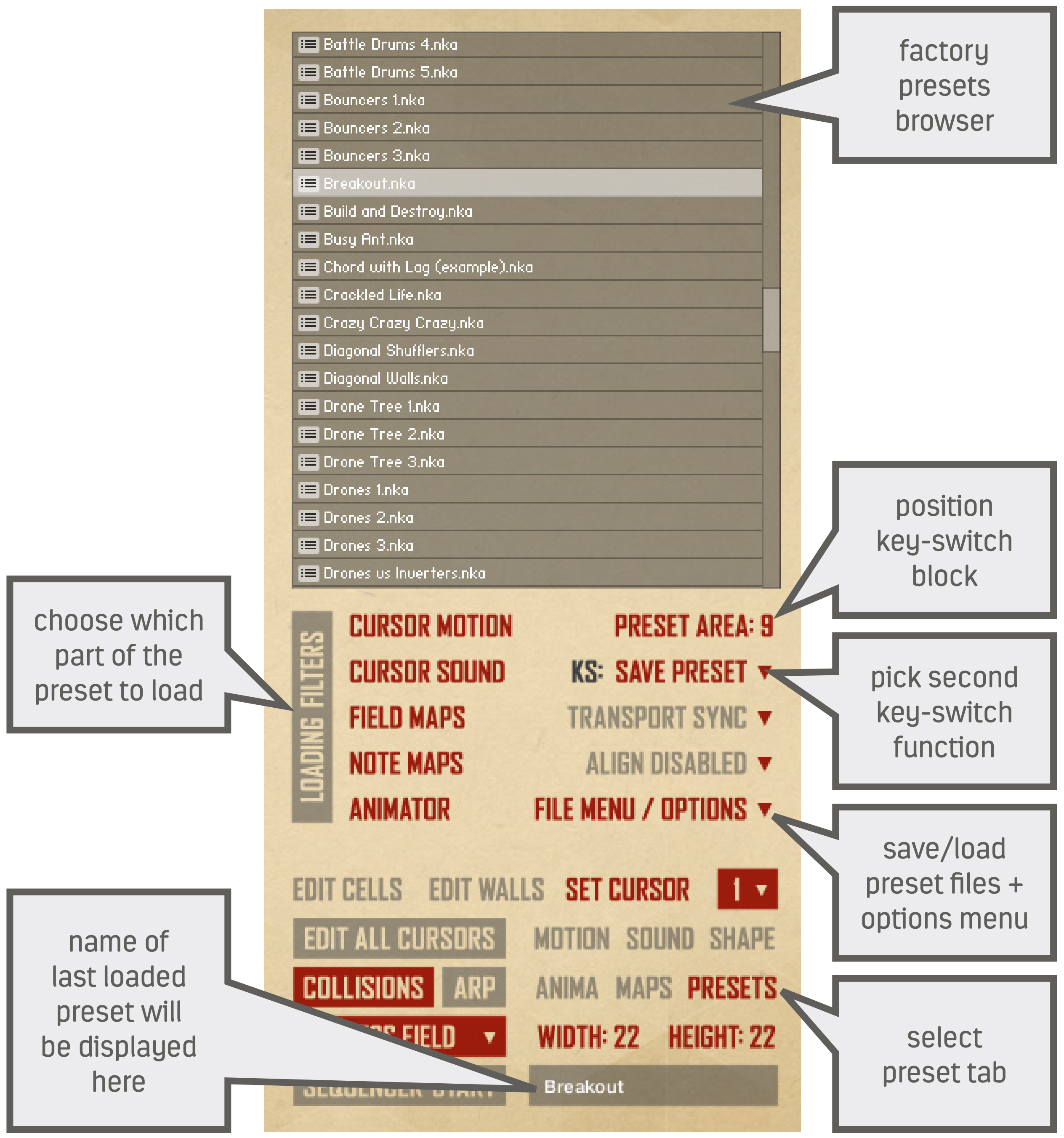
Cracklefield comes with a variety of presets available via preset browser. To load factory preset double click the preset name. The preset browser can also be navigated by cursor keys – single click on browser to select it, then you can use up/down cursor keys to navigate the list and enter key to load preset. Pressing letter keys will jump through the list alphabetically.

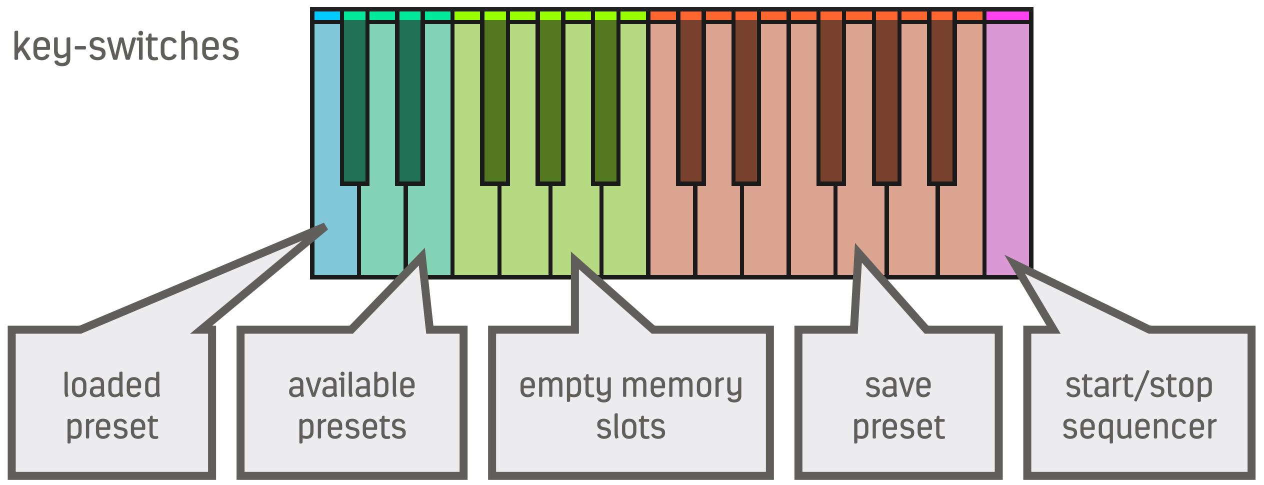
Internal memory can only be accessed by key-switches (midi keys or virtual keyboard). There are two key-switch blocks, use red coloured keys to save a preset, use green coloured keys to load a preset. Load preset block keys vary in colour shade, yellow-green keys indicate an empty memory slot, darker green keys indicate that a preset has been saved and assigned to the key. Cyan-coloured key indicate last loaded/saved slot. There is additional magenta coloured key-switch, which automates sequencer start/stop button.

All of internal memory is being saved within instrument patch file (NKI) and a DAW project. It is also being saved within snapshot file.

The key-switch block can be placed anywhere on MIDI keyboard (it will be aligned to octaves), to position key-switches in convenient keyboard area, use PRESET AREA controller (acts as vertical slider). You can also reverse order of key-switch blocks (save block first, then load block), to do so, use KS (key-switch) drop-down menu and pick last option "swap keyswitch blocks". To remember the setting re-save instrument patch.

The second key-switch block, used to save presets, can be also used for different functions – change scale/pattern key, or program scale/pattern. To change key-switch functionality, use KS drop-down menu.


Battle Drums 4.nka
Battle Drums 5.nka
Bouncers 1.nka
Bouncers 2.nka
Bouncers 3.nka
Breakout.nka
Build and Destroy.nka
Busy Ant.nka
Chord with Lag (example).nka
Crackled Life.nka
Crazy Crazy Crazy.nka
Diagonal Shufflers.nka
Diagonal Walls.nka
Drone Tree 1.nka
Drone Tree 2.nka
Drone Tree 3.nka
Drones 1.nka
Drones 2.nka
Drones 3.nka
Drones vs Inverters.nka



Preset contents are divided into four parts, you can select which part to load when loading a preset, using five switches grouped next to LOADING FILTERS label.

CURSOR MOTION part includes: cursor on/off state, position, direction, diagonal mode, cursor speed, rate, type and mode, cursor offset, state of sequencer speed controller, field width and height, state of collisions switch. Enabling this part also makes sequencer restart the counter when loading a preset.

CURSOR SOUND part includes: cursor sound set, cursor volume, pitch and pan, cursor mapping mode and trim/add/chord size modifier, cursor attack, hold and tail parameters, cursor lag, cursor arp options and state of arp switch, hold and lag randomise ranges, volume, pan and pitch randomise ranges, cursor selection (which cursor is selected – as this option determines which sound can be played manually).

FIELD MAPS include: field map and wall map, including hidden part, outside of selected dimensions.

NOTE MAPS include: note range, note pattern/scale and key, note pattern distribution mode, velocity bank, velocity distribution mode, velocity random range, user recorded note and velocity maps.

ANIMATOR covers all animator related options, configuration of individual animators, state of animator on/off button, animator speed rate.

Note that these switches only affect preset loading function (as well as random preset generator). Saving a preset always include all parts.

Also, note that these switches affect UNDO functionality, as UNDO is based on preset load/save function. If you switch one of them off, calling UNDO will not recall all the data.

To quickly enable loading just a single preset part hold ALT key and click on given switch. To quickly enable all switches hold SHIFT key and click on any filter switch.

To load presets seamlessly, while sequencer is playing, you can activate ALIGN function. It will delay the moment of loading the preset, to align it with selected time frame (32nd note, 16th note, 8th note, quarter note or whole note). In plug-in mode (as opposed to standalone Kontakt mode), ALIGN function will also delay sequencer start to align the sequence with transport (position in project song).

Additionally you can make the sequencer start whenever play button is pressed in DAW (as well as stop with stop button), by enabling TRANSPORT SYNC option.

Presets can be also saved to a file. To do so, use 'save preset' and 'load preset' options from FILE MENU. Default preset save/load location is Data folder, placed in the instrument path root folder. You can pick any folder you like, but Kontakt will always point you to Data folder first.

Saved files are in the same format as factory presets. You can add presets to factory set, by saving or copying them to "presets" folder. However, to refresh preset list in the browser, you need to close and reload the instrument.

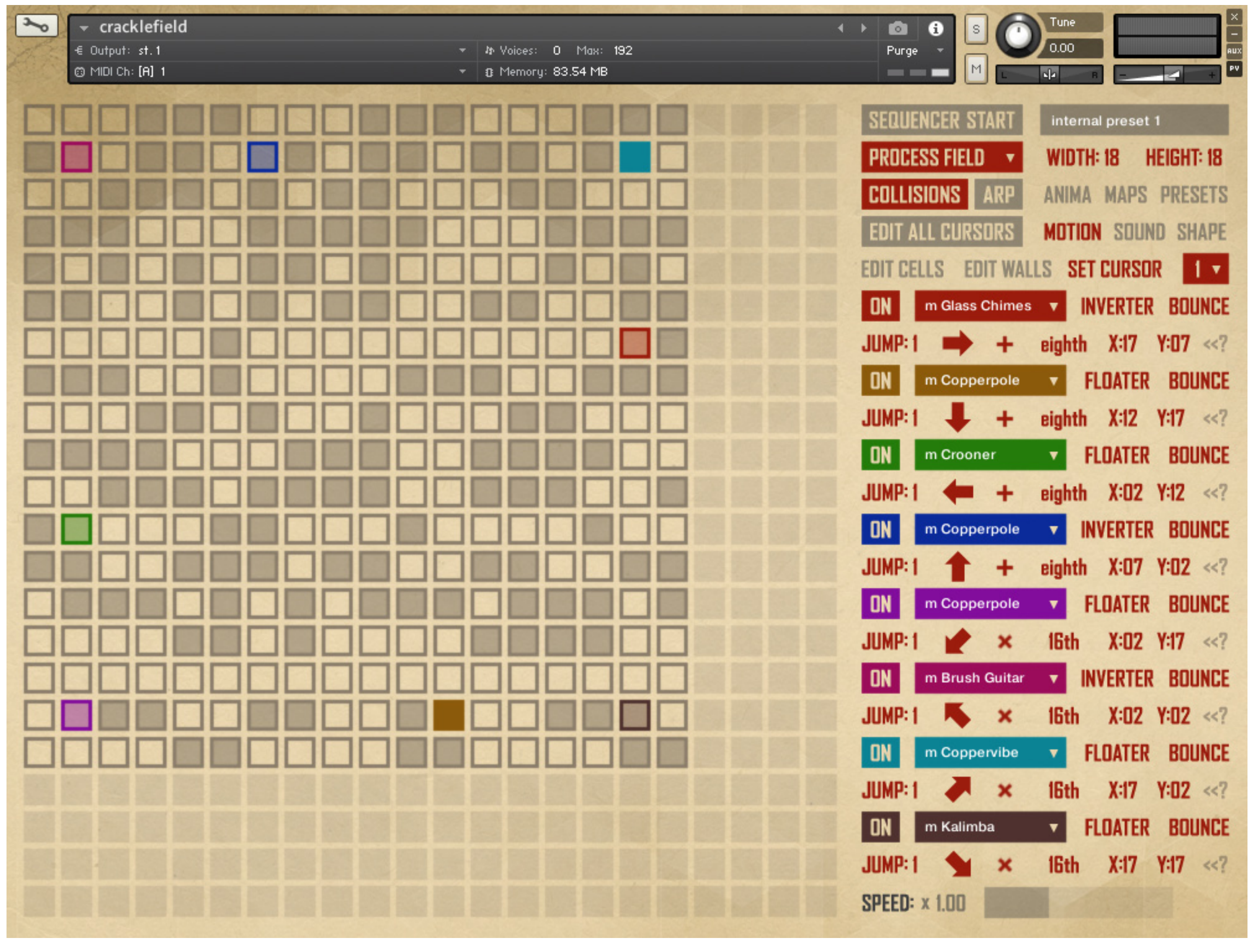
Note that 'presets' folder is expected at the same path, as loaded NKI file. If you save variations of the instrument, keep them in the same as the original patch. Otherwise preset browser will be empty.

The name of last loaded factory preset, or last saved/loaded internal memory slot, will be displayed on sequence counter label, next to sequencer start button when the sequencer is stopped. Displayed preset name will be cleared when resetting the instrument ('full reset' option in 'process field' menu), or when loading/saving preset to file. It can be also cleared manually, by picking 'forget loaded preset name' option from FILE MENU / OPTIONS drop-down menu.

Alternative UI layout

User interface in Cracklefield is fairly large, taking 1000x770 pixels, including script tabs. In some hosts, or configurations it can happen that not all of it is visible at once. In such case, as well as for purely aesthetic reasons, you can switch sequencer to alternative layout. In this configuration sequencer on/off switch block is placed at the top of the interface area. It will give you access to more commonly used controllers without scrolling.

To switch UI layout, pick 'change UI layout' option from FILE MENU / OPTIONS menu in PRESETS tab. You will need to save the instrument patch to remember the setting.



Controller options

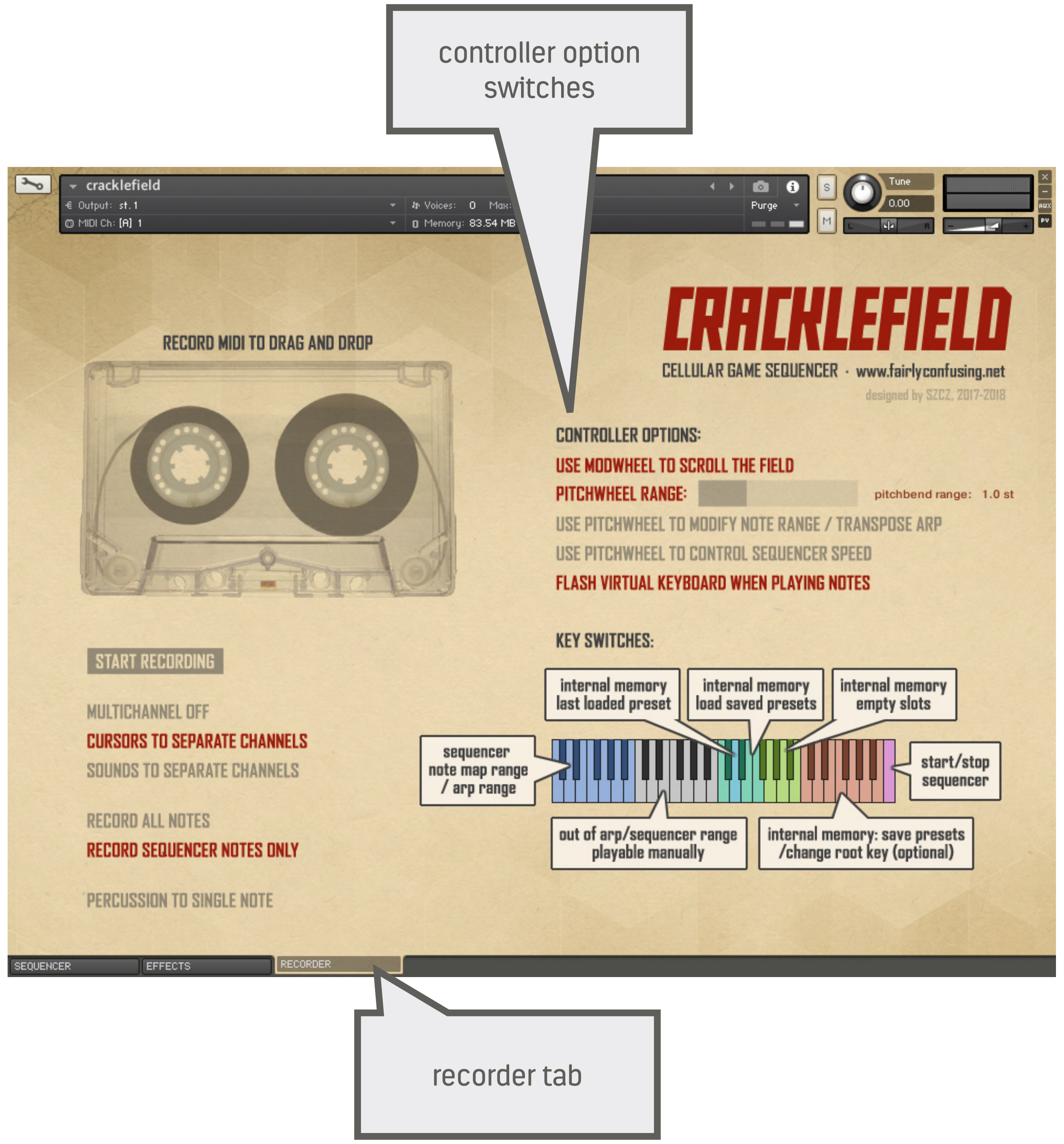
Mod-wheel and pitchbend-wheel can be used to interact with the instrument, while sequencer is playing. You can configure these controllers in RECORDER tab.

USE MODWHEEL TO SCROLL THE FIELD – this option is activated by default, moving the modwheel will scroll the field map up or down. Field contents will be wrapped around the field edge (moving the field upward, moves the top line to the bottom of the field). The entire field is used, if dimensions are trimmed with width and height controllers, the data will move in and out of the hidden field area. Note that scrolling only affects the cell's state (filled/empty), note and velocity maps are not being scrolled. It's interesting way to interact with the sequencer, but if you rather wish to use modwheel for another purpose, it can be turned off.

USE PITCHWHEEL TO MODIFY NOTE RANGE / TRANSPOSE ARP – normally pitchwheel acts as pitchbend controller. When this option is enabled, you can use pitchwheel to move the note map range up/down, dynamically re-writing the note map. In effect, it is transposing all notes played by cursors using note map. In ARP mode, this option will simply transpose generated notes.

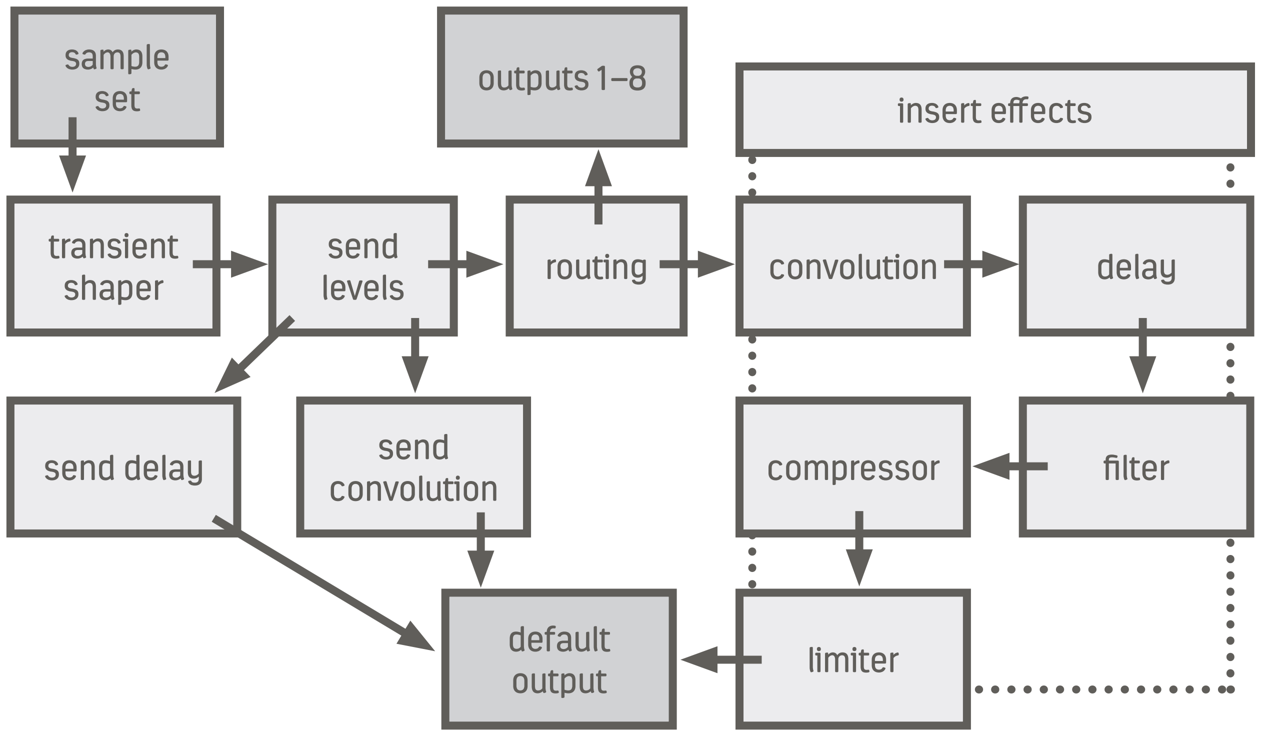
PITCHWHEEL RANGE – you can use the horizontal slider to set pitchbend effect range, or (when alternative mode is enabled) arp transpose range.

FLASH VIRTUAL KEYBOARD WHEN PLAYING NOTES – as you can observe, the sequencer colours the keys on Kontakt's virtual keyboard, when playing notes (only for sequencer generated notes). It can be useful feature, to monitor the setup, but it can also be a little annoying. It can be turned off with this switch.


controller option switches
cracklefield
CRACKLEFIELD
CELLULAR GAME SEQUENCER · www.fairlyconfusing.net
RECORD MIDI TO DRAG AND DROP
CONTROLLER OPTIONS:
USE MODWHEEL TO SCROLL THE FIELD
PITCHWHEEL RANGE:
pitchbend range: 1.0 st
USE PITCHWHEEL TO MODIFY NOTE RANGE / TRANSPOSE ARP
USE PITCHWHEEL TO CONTROL SEQUENCER SPEED
FLASH VIRTUAL KEYBOARD WHEN PLAYING NOTES
KEY SWITCHES:
START RECORDING
MULTICHANNEL OFF
CURSORS TO SEPARATE CHANNELS
SOUNDS TO SEPARATE CHANNELS
RECORD ALL NOTES
RECORD SEQUENCER NOTES ONLY
PERCUSSION TO SINGLE NOTE
internal memory last loaded preset
internal memory load saved presets
internal memory empty slots
sequencer note map range / arp range
start/stop sequencer
out of arp/sequencer range playable manually
internal memory: save presets /change root key (optional)
SEQUENCER
EFFECTS
RECORDER
recorder tab

Effects and multi-output

There is a set of audio effects in Cracklefield: compressor, limiter, transient shaper, basic filter, two convolution reverbs and two delays (send and insert). Audio signal path is outlined below.

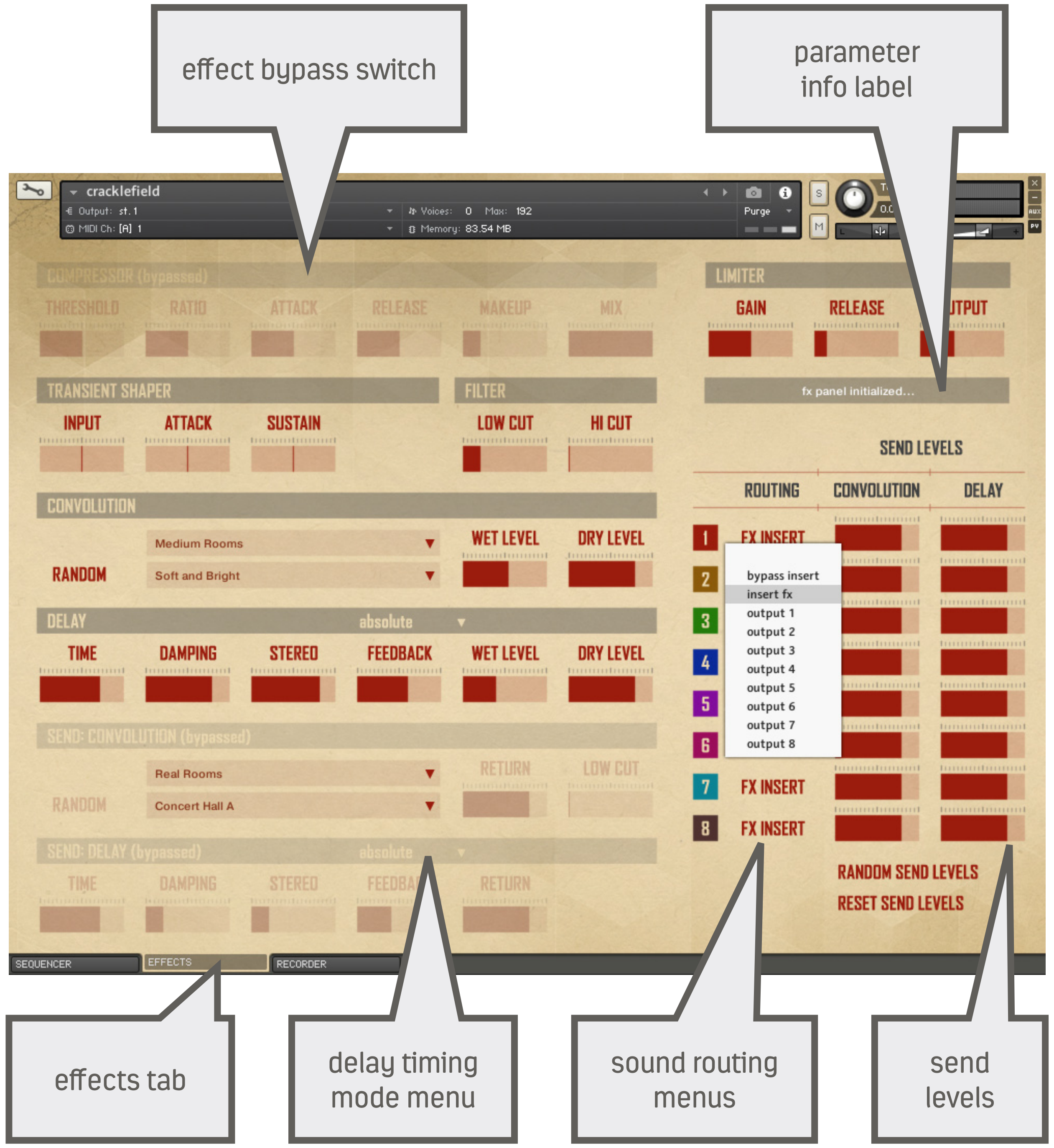


Any effect can be switched to bypassed mode by clicking on the effect name label (whole gray bar is bypass switch).

While moving a controller, the parameter value will be displayed on info label below limiter.

Convolution reverbs use Kontakt's factory impulse-response library. You can use drop down menus to select an impulse, or press RANDOM button to let the machine select a random one.

There are send level controllers for each cursor (numbered and colour coded as cursors). However, as the instrument is built on one sample set per group principle, controllers for cursors with the same sample set will be linked. If you assign the same sample set to all cursors, moving any send level controller, will affect all send levels of the same type. You can randomise send levels by pressing RANDOM SEND LEVELS button.

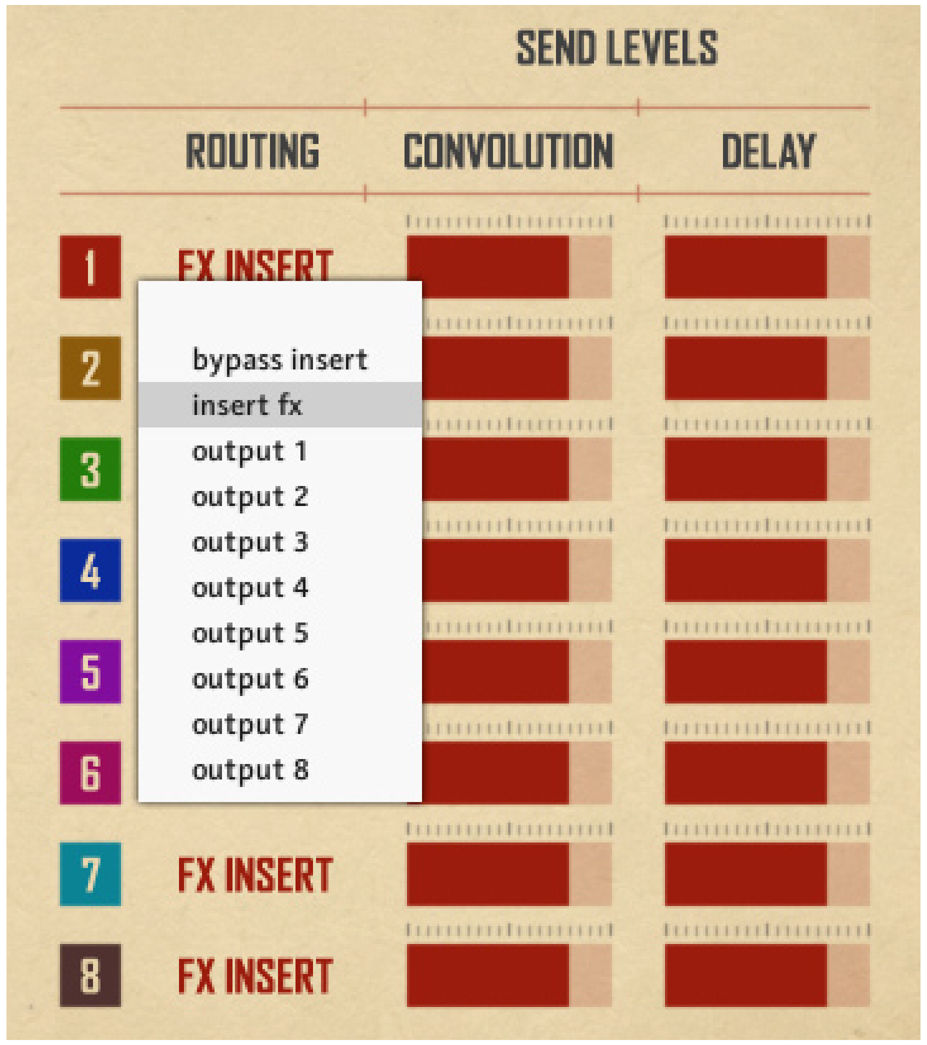


Delays can be switched between absolute and tempo synchronized timing modes, using drop-down menus. TIME controller sets actual delay time in absolute mode, or multiplier of selected note fraction for tempo synchronized mode.

To tame possible excessive low-end generated by convolution, you can use LOW CUT controller (high pass) for send convolution reverb, or insert filter's LOW CUT for insert convolution (which affects whole chain).

If you want to use different effect chains for different sample sets outside Kontakt, you can configure Cracklefield to multi-output. You can pick specific output number from ROUTING drop-down menus. Output 1 to output 8 are Kontakt output slot numbers, actual routing depends on Kontakt configuration. Selecting INSERT FX, will route signal to insert effect chain and then to default Kontakt output, selecting BYPASS INSERT routes signal directly to default output, skipping all insert effects. Note that routing signal to a specific output, also skips insert effect chain.

Transient shaper can be used to modify balance between initial percussive part of a sound and the sustained part. Use ATTACK and SUSTAIN controllers to make percussive and sustained parts of the sound louder or quieter. Note that transient shaper is not a part of insert effect chain. It is applied at the beginning of the chain, so it affects what is being passed to send effects. Also it is being applied to sound routed to different outputs (unlike insert effects).



MIDI recorder

Cracklefield comes with a simple midi recorder, outgoing notes can be recorded and then exported to a file or directly to DAW by drag'n'drop.

Sometimes it can be more convenient to record a clip inside the instrument and drag it into the DAW, than to configure recording a track from Kontakt's MIDI out. But the real reason for including the recorder, is that it can create multichannel clips, where different cursors are recorded to separate MIDI channels.

To start the recording, use START RECORDING button. There is no need to synchronize it with sequencer start, after pressing the button, recorder will wait for a midi note to start actual recording. When recording is in progress, recorder will display a counter, showing how many notes can still be recorded and the recording time thus far. Recorder capacity is 100000 events, that is, 50000 notes (note on + note off message). It should handle at least 20 minutes of a dense sequence.

Press START RECORDING button again to stop recording. When finished, click on cassette image and drag it out of Kontakt window to create clip file. In fact, you can drag and drop MIDI clip anytime, even when the recorder is running.

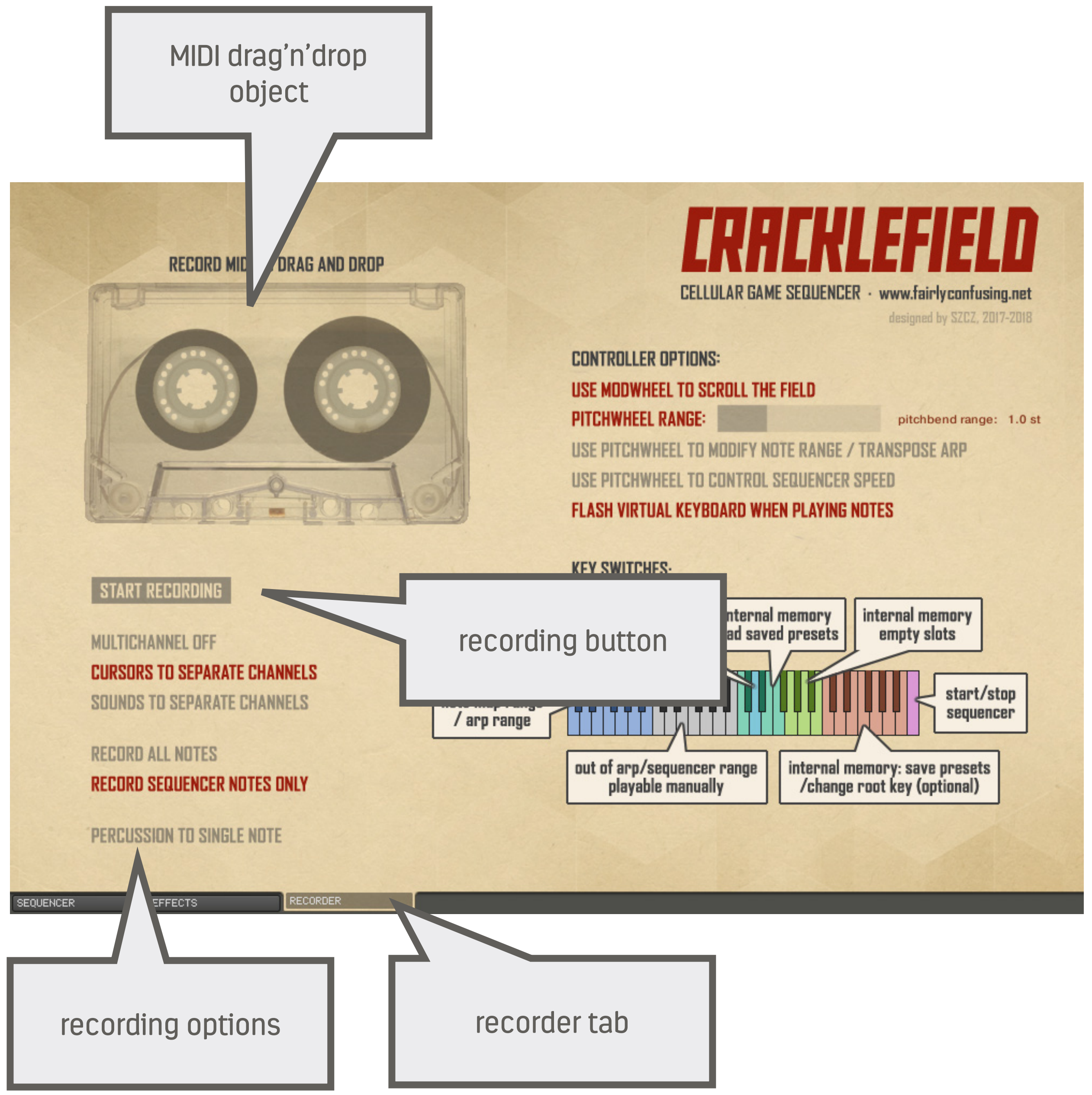
Note that starting the recording erases MIDI buffer, so previous recording is lost. There is no undo here, so be careful. Also, remember that the recorder only records MIDI notes, any other data is ignored.

Multichannel recording options:

CURSORS TO SEPARATE CHANNEL – in this mode each cursor will output notes to separate channel.

SOUNDS TO SEPARATE CHANNEL – channels will be sorted by sample sets, different cursors with the same sound set selected will be put in one channel.

MULTICHANNEL OFF – all notes go to single channel.


MIDI drag'n'drop object
RECORD MIDI DRAG AND DROP
CRACKLEFIELD
CELLULAR GAME SEQUENCER · www.fairlyconfusing.net
designed by SZCZ, 2017-2018
CONTROLLER OPTIONS:
USE MODWHEEL TO SCROLL THE FIELD
PITCHWHEEL RANGE:
pitchbend range: 1.0 st
USE PITCHWHEEL TO MODIFY NOTE RANGE / TRANSPOSE ARP
USE PITCHWHEEL TO CONTROL SEQUENCER SPEED
FLASH VIRTUAL KEYBOARD WHEN PLAYING NOTES
KEY SWITCHES:
START RECORDING
recording button
internal memory load saved presets
internal memory empty slots
MULTICHANNEL OFF
CURSORS TO SEPARATE CHANNELS
SOUNDS TO SEPARATE CHANNELS
/ arp range
start/stop sequencer
RECORD ALL NOTES
RECORD SEQUENCER NOTES ONLY
out of arp/sequencer range playable manually
internal memory: save presets /change root key (optional)
PERCUSSION TO SINGLE NOTE
SEQUENCER
EFFECTS
RECORDER
recording options
recorder tab

More recording options:

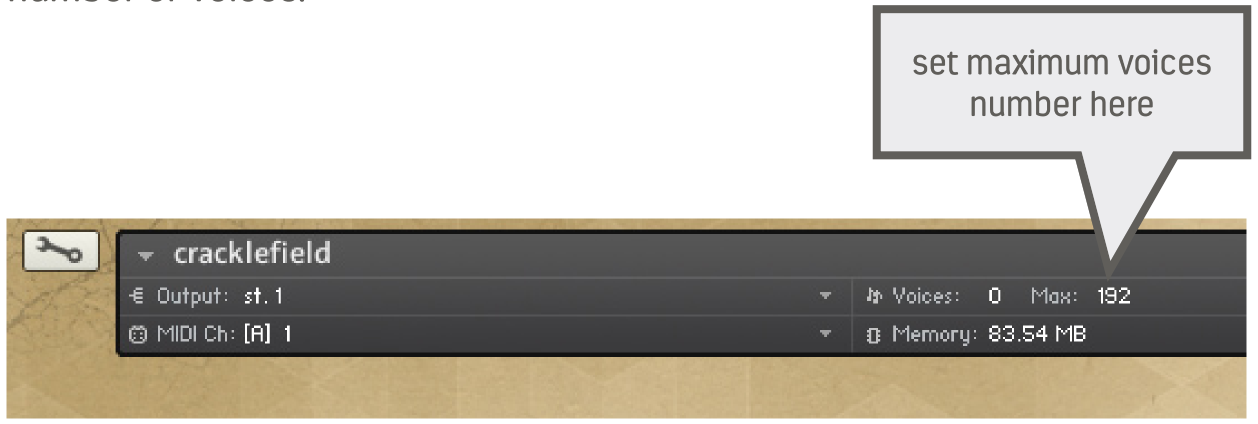
RECORD ALL NOTES / RECORD SEQUENCER NOTES ONLY – choose, if you'd like to record only notes generated by the sequencer, or manually played notes as well.

PERCUSSION TO SINGLE NOTE – percussive sound sets in Cracklefield use different note values for round robins. If you'd like to play recorded clip in another instrument, it would make sense, if each percussive instrument generated a specific note value (different for each sound). Use this option to limit percussive instruments output. Note that the instrument makes the distinction between melodic and percussive sound sets, this option only affects percussive ones, so the two types can be mixed.

Performance

Cracklefield doesn't require much processing power compared to other instruments, as it doesn't use many effects and modulators. For this reason, I thought I could afford to set somewhat generous polyphony limit. Instrument maximum voices is set to 192, which I haven't yet managed to reach using provided sounds.

If there is a need to reduce CPU consumption, try reducing the maximum number of voices.


set maximum voices number here

cracklefield
Output: st. 1
MIDI Ch: [A] 1
Voices: 0 Max: 192
Memory: 83.54 MB

Changelog

1.01 update:

– In rare circumstances Cracklefied could generate a note with zero length. When a sample with loop was used, it would result in never ending note. There is now minimum note length limit set to 1/100 second.

– Cursor rate randomiser could set cursor offset to half note triplet. While this setting works fine, there is no such option in the selector menu, resulting in missing graphics glitch. Half note triplet has been ruled out as random offset value.

1.2 update:

– Added field animators, see page 32.

– New cursor type: beacon, see page 13.

1.3 update:

– Kontakt version requirement is now raised to 5.8.1. This Kontakt version has doubled the limit of available controllers per script, which is a great benefit for Cracklefield. I had run out of switches and buttons, as creating programmable field consumed many controllers of this type. When having no button controllers left to use, I had to reach for alternatives, using drop-down menus and sliders instead of buttons for some other controllers, which had impact on functionality. In version 1.3 those controllers are converted to regular buttons. It applies to 'move/bounce' switches in motion tab, 'move maps' switches in animator tab, note pattern and key controllers in maps tab and some UI tab switches.

– On a request, there are new controllers for manual editing or tweaking of single cells in custom maps. You can hide/show them using 'tweaks' button in MAPS tab. See page 26.

Changelog

1.4 update:

– Added progression generator, pattern/chord progressions can now be generated internally in Cracklefield, see page 18.

– There are new presets exploring the progression generator functionality, new presets have "PSQ" prefix. Also several old presets have been reworked to use progression.

– Tweaked random preset generator, the randomly generated presets should be simpler more often, possibly giving more useful results. The generator will affect the animator, reset it or occasionally set up a moving field. The generator will now also respect preset loading filters.
See page 6.

– Fixed scale key button set not being updated on certain occasions.

– The seed for random event generator will now be reset to a derivative of field checksum every time transport starts (you press play or record in DAW). This way any random event sequence in the project will be synchronized with field contents and each time you render a saved song it would repeat the very same progression of pseudo-random events.

– Small change in main sequencer queue, the animator will be now applied 1/3 sequence step ahead of other functions (with the exception of the first step). As the animator transformations can be CPU intensive, especially if you make it play several life games at one step, coupled with progression generator, it could occasionally cause audio dropdowns. Now that animator is processed a small while ahead of other functions it should make such situations less likely to happen.

End of file

I'd like to thank James Michael Wolk for beta testing and creative input which influenced the shape of the instrument in many ways.

If you'd like to hear about possible updates and news, subscribe to the blog here: http://waveforms.fairlyconfusing.net

I will post updates and new products there. Alternatively you can follow my Facebook page or subscribe to Youtube channel, see the links below. I do not have a mailing list.

Have fun with Cracklefield.

This document ends here.

www.fairlyconfusing.net

marcin@fairlyconfusing.net

www.facebook.com/szcz.audio.adventures/

www.youtube.com/c/SzcZ