Chord Progressor

1.5

Manual

Content:

[1] The concept behind the creation of ChordProgressor

While there already is an abundance of great music software available, some of which exceeds ChordProgressor by far regarding the scope and complexity, it is a proven concept in creating arts to restrict oneself to a certain set of tools to avoid distraction and aimless wandering through thousand and more of options.

Despite its potential for usage in musical education the basic intention behind ChordProgressor was to create a specific tool to generate modulated & rhythmically arpeggiated chord progressions, which would control any of the cool sound generators at hands out there, whether they be made of hardware or software.

Since every music creator has a very personal taste and preference when it comes to chords, arpeggios and rhythms, the tool was envisioned to include a kind of a library of all previously created musical pieces, so one could reuse these in a different context later on as well as in different combinations. In addition to the more common feature set of composition tools like this, the arpeggiator part was devised in a very generic way to enable the composer to use any combination of chord notes at a given step.

Since ChordProgressor does not aim to replace your beloved DAW (DigitalAudioWorkstation: either an AudioSequencer or an external Hardware synth with sequencer functionality) of choice, but is intended as a standalone tool to prepare loops being used inside the DAW (thought it may be run alongside the DAW in MIDI-beat-clock sync mode), it comes with some MIDI file export options, so the generated MIDI data can be used by pretty much any digital audio soft- and hardware on the market.

Even though originally tailored to my specific production needs, I hope ChordProgressor finds fellow music creators which will be inspired and empowered by it while cooking up new song pieces.

René Laudi, August 2014

[2] Installation

All versions of ChordProgressor come as a compressed ZIP archive, which either contains an installer (for Windows operating systems) or the application (for all other operating systems, which usually contain all other neccessary operating components).

[2.1] Mac OS X

The Mac OS X ZIP archive contains only the application and this manual inside a "ChordProgressor" folder. Please unpack the ZIP archive and move the unpacked folder to your "Application" folder or any other place to your liking on your harddrive.

On ChordProgressors 1st launch it creates a folder named "ChordProgressorData" next to the CP application, which contains all library presets plus any newly created CP data by you.

[2.2] Windows XP / Vista / 7 / 8 installation

The windows installer package comes as an zip compressed executable, which guides you though the installation process after being unzipped and started. Besides the ChordProgressor application and this manual it contains a complete Java runtime engine to ensure maximum compatability.

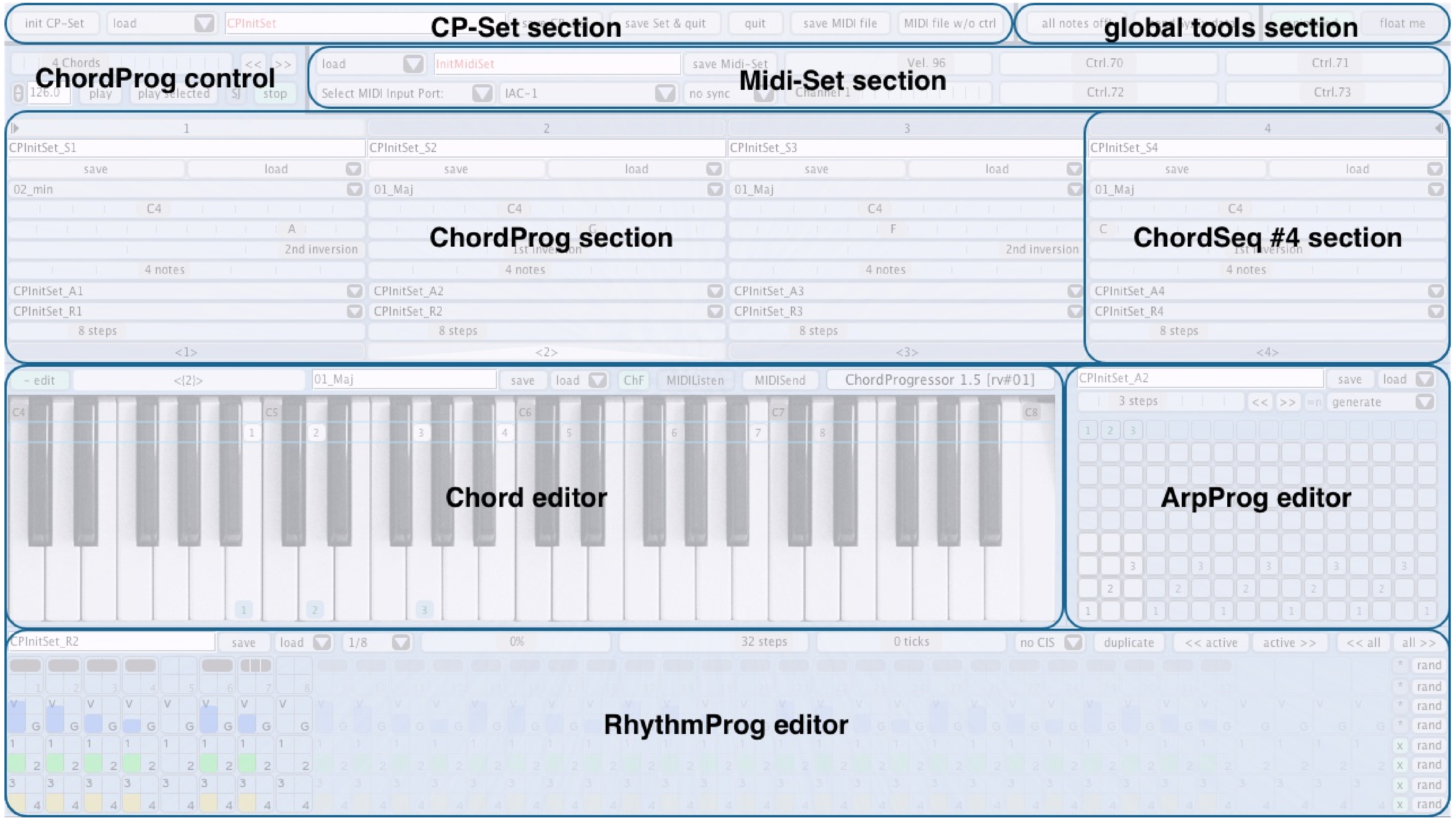
Since Windows inhibits write access to the application folder, ChordProgressors creates a folder named "ChordProgressorData" in your user data folder on it's 1st launch, which contains all library presets plus any newly created CP data by you.

[2.3] Linux / Unix installation

The Linux / Unix ZIP archive contains only the application launcher and this manual inside a "ChordProgressor" folder. Please unpack the ZIP archive and move the unpacked folder to your "Application" folder or any other place to your liking on your harddrive.

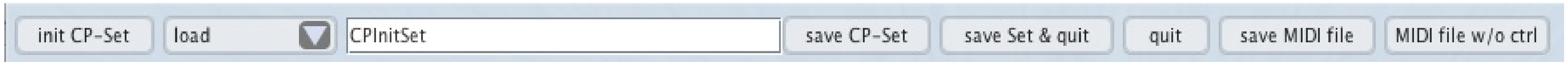
On ChordProgressors 1st launch it creates a folder named "ChordProgressorData" next to the CP application, which contains all library presets plus any newly created CP data by you.

[3] The building blocks of ChordProgressor



[3.1] ChordProgressors Set (CP-Set) section

At the top left of the ChordProgressor (CP) window you'll find all functions related to the ChordProgressor data set (CP-Set) described from left to right in the following:



init CP-Set: if clicked with the left mouse button CP will set up a generic 4 chord sequence as a starting point for you. When you click on the edit button the edit mode will be prepared with MultiEdit on so you can modify the arpeggio and rhythm for all chords at the same time for an easy start.
If clicked with the right mouse button this loads an inital set comprised of a plain pop 4 chord progression with some changing arpeggio & rhythm style on it. This is the same set, you'll see on the very first start of CP. You may change this one by copying any CP-Set created by you from the ChordProgressorData/CPSets folder next to the CP application to the ChordProgressorData folder itself and renaming it to CPInitSet.cps (respectively CPInitSet if your system is set up to hide all file extensions) after deleting the original CPInitSet.cps.

load: in the opening list panel you'll find any CP-Set you have saved before. Please note that although the list panels in CP are modal (means you can't change anything outside the list panel while it is open) the sequencing engine will keep on running if you started it before opening any list panel, so you can have a peek listening to what you choose to load. If you right-click on the load button, you'll load a chord progression from the CP-Library, which just contains references to chords, arpeggios and rhythms via their names as stored in the CP-Library, whereas a CP-Set contains the complete data in one package. You'll find more information on this subject further down.

the text field: contains the name of the loaded CP-Set (respectively the chord progression from the CP-Library) or if you enter a new name the name with which the CP-Set will be saved when you click on {save CP-Set}. Please be advised that a newly edited name needs to be confirmed by either pressing {enter}, {tab} or the {save CP-Set} button. (This rule applies to all textfields in CP btw.)

save CP-Set: pressing this button you'll save all the current CP data to a CP-Set with the name entered in the text field plus also the chord progression as reference to its parts consisting of chordSequences, chords, arpeggios and rhythms to the CP-Library.

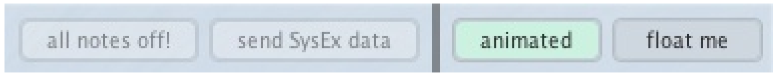
save Set & quit; quit: your fast way out of CP! Either with saving the current CP-Set or without, which means you discard all changes since you have save your CP-Set last.

save MIDI file; MIDI file w/o ctrl: with these two buttons you have 4 different flavours to export your finely finished chord progression MIDI loop to be used in other applications of your choice:

1) left click on {save MIDI file}: save your complete CP-Set composition as a MIDI file to a location of your choice ready to be imported into e.g. your DAW
2) right click on {save MIDI file}: save the currently selected ChordSequence as a MIDI file
3) left click on {MIDI file w/o ctrl}: save your complete CP-Set without any controller data (no Ctrl.1 - Ctrl.4, notes only) as a MIDI file ready to rock your DAW
4) right click on {MIDI file w/o ctrl}: save the currently selected ChordSequence as a MIDI file without any controller data (no Ctrl.1 - Ctrl.4, notes only)

[3.2] ChordProgressor Global Tools section

At the top right of the ChordProgressor (CP) window you'll find some useful global functions:



all notes off!: this sends a rude bulk of 128 note offs to the selected MIDI output port to shut up any pending notes. This button is disabled as long as there is no MIDI output port selected.

send SysEx data: in case you've got an *.syx file containing MIDISysEx sound data, you can send it to your synth with this function. As long as there is no MIDI output port selected this button is disabled.

animation: this button has 3 states: {animated} (green, left click), {animation} and {0 animation} (red, right click) with the following functions:

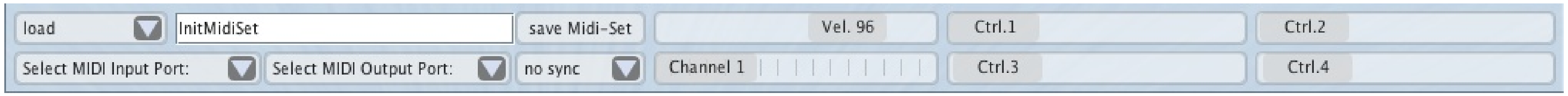
1) {animation} (blue-gray): if not in edit-mode the editors follow the chord progression
2) {animated} (green): as above plus animation of each individual note as it is played
3) {0 animation} (red): no animation except for the moving of the ChordSequence pointer

In case you've got an up to date processing-power-monster at your command you can just stick it to mode 2), in case you still use your trusty vintage computer gear, you've got some control over the burden on your machine here, as the graphics toll may interfere with the MIDI timing performance and CP's responsiveness depending on your system.

float me: when you activate this button, ChordProgressor will always stay on top of things (well, except for even more floaty plugIn windows winning the match) <nerd hint #1: there is a hidden preference setting in the file "ChordProgressorData/ChordProgressor_prefs.xml" named "noCaptionBar" which, if set to 1, will remove the caption bar of CP after restarting CP, so you gain a little more screen real estate.> <nerd hint #2: after applying nerd hint #1 you may ask youself how you gonna move the window from now on -> click, hold and drag anywhere in the CP windows where there is no control element.> <general advice #1: if you manually alter any data in the directory ChordProgressorData esp. to funky values, please don't expect CP to take that with grace in any case - and don't say you haven't been warned.>

[3.3] ChordProgressors Midi-Set section

Most of the lower right top of the CP window is covered by its Midi-Set functionality:



A CP Midi-Set contains all informations which are related to your sound device, which could be a hardware synthesizer, a standalone software synthesizer running alongside CP or your DAW containing a synth plugIn which is played via the MIDI In of the DAW.

It is intended to keep the sound specific controller definition data separate from the compositional data, so once you've defined some Midi-Sets for your favourite sound devices in your studio, you can concentrate on juggling the notes and modulations. Therefor it is advisable to name the Midi-Sets according to your synths, etc. and possibly also according to the matching sound setup of your synth.

load: not much of a surprise here, this loads a previously defined and saved Midi-Set of yours from the CP-Library. Please note, that loading a CP-Set also automatically loads the Midi-Set saved inside the CP-Set, thus overriding whatever you have set up here. So you might want to keep an eye on saving a Midi-Set right after you've finalized it to the CP-Library with the next button to the right, so you can easily use it on different CP-Sets, too.

the text field: as with the CP-Set text field you enter the name of the to be saved Midi-Set here - keep in mind to press enter, tab or {save Midi-Set} to store it.

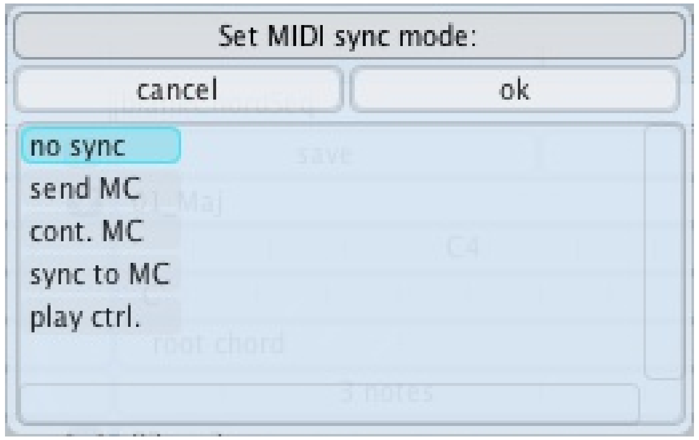
save Midi-Set: saves all the settings in this area to a Midi-Set named according to {the text field} to the CP-Library

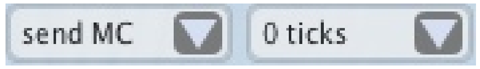
Some general information about the user interface controls of CP for a start: To save screen space, CP uses only a few labels, but supplies you with helpful informations via tooltips which show up as soon as you pause the mouse pointer for a little while over a control. Additionally all slider controls have a selection list mode, too, which can be accessed via right mouse click on the slider. The range of some controls may depend on the setting of other controls, so you might want to have an eye on the whole picture while changing values, to get accustomed to the dependencies, which will be pointed out in the particular information per control, too.

Select Midi Input Port: CP scans all MIDI ports registered in your system at its startup and one of these can be selected from the resulting list here. The first entry in the list switches the MIDI input of CP off.

Select Midi Output Port: same as above for the MIDI output port. Please note, that changes in your MIDI setup while CP is running might need a restart of CP to update these lists for inclusion of new devices.

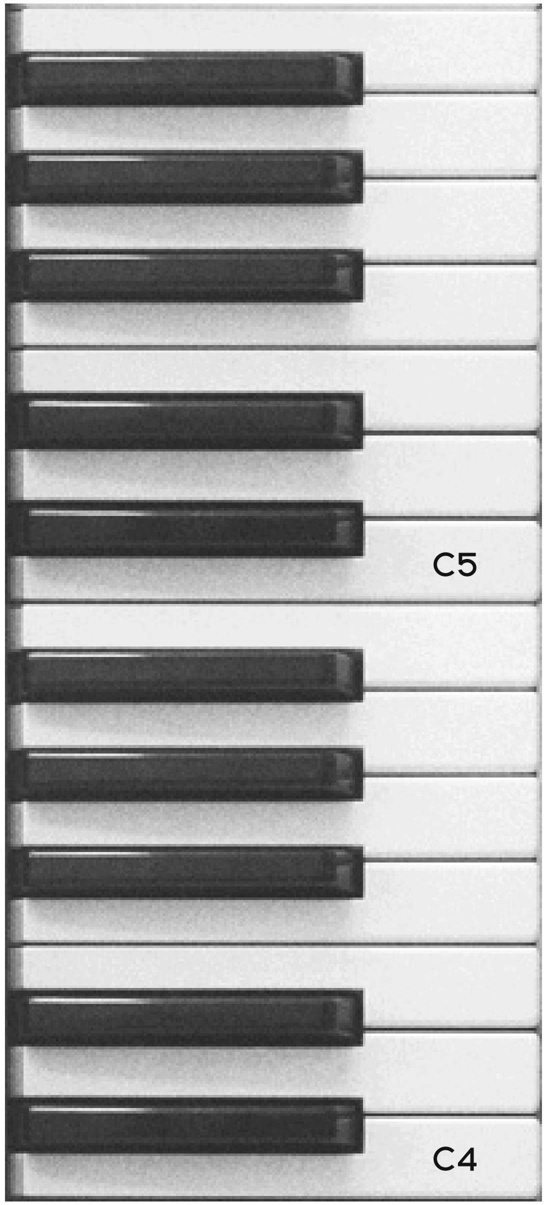
Midi Sync Menu: here you can choose one of the following MIDI synchronization options:





1) {no sync}:	all external control options via MIDI are turned of
2) {send MC}:	send MIDI beat clock inbetween start & stop
3) {cont. MC}:	send continous MIDI beat clock independent of start & stop
4) {sync to MC}:	synchronize CP to incoming MIDI beat clock and start & stop MIDI commands
5) {play ctrl.}:	control CP via your MIDI keyboard (please see table next page)

If you select {send MC} or {cont. MC}, an additional list shows up, so you can set a MIDI clock (pre-)delay



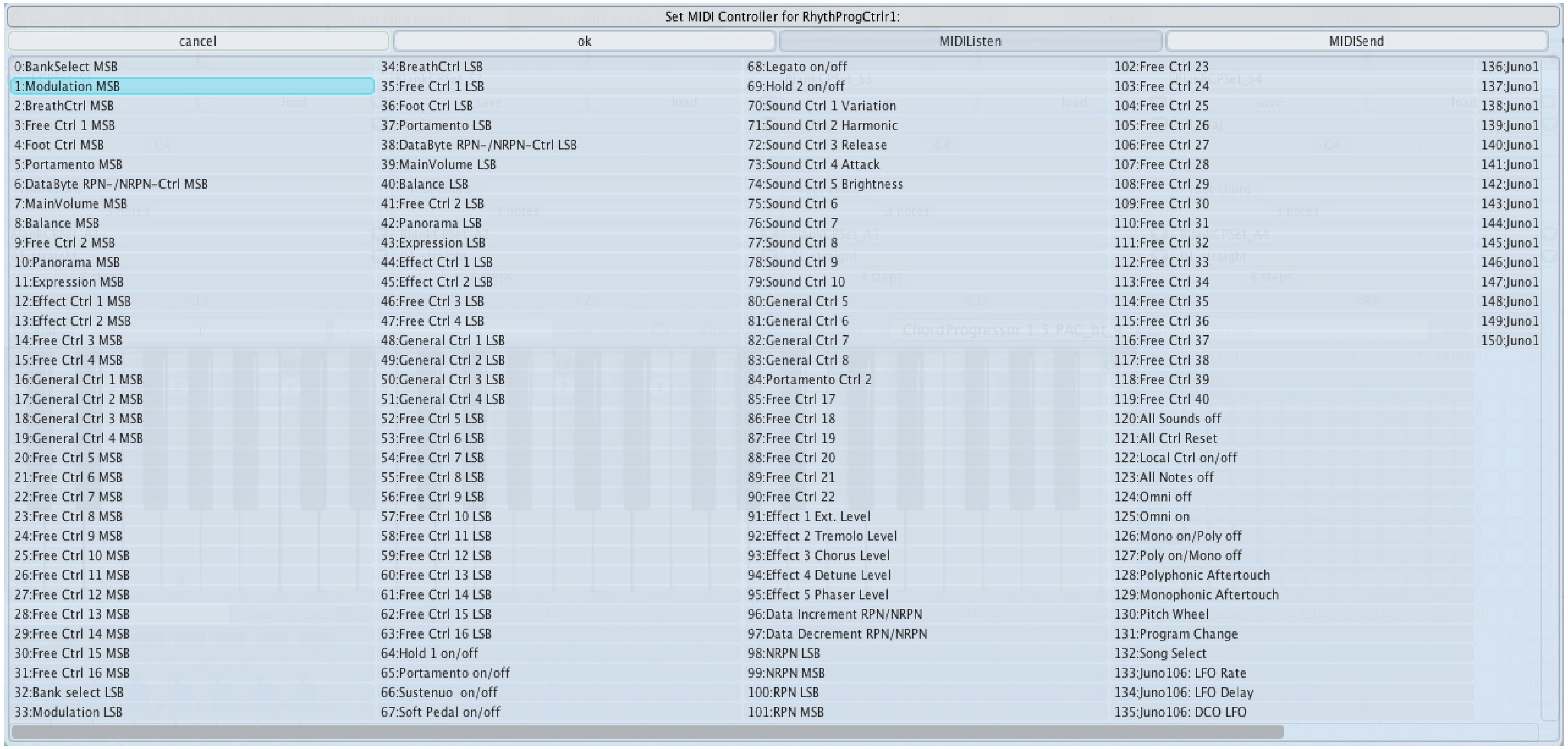
Play Single Step backward of last used Play Mode
unused
Play Single Step forward of last used Play Mode
Select Next ChordSequence
Stop
Select Previous ChordSequence
Play Single ChordSequence
Play All
Select ChordSequence <16>
Select ChordSequence <15>
Select ChordSequence <14>
Select ChordSequence <13>
Select ChordSequence <12>
Select ChordSequence <11>
Select ChordSequence <10>
Select ChordSequence <9>
Select ChordSequence <8>
Select ChordSequence <7>
Select ChordSequence <6>
Select ChordSequence <5>
Select ChordSequence <4>
Select ChordSequence <3>
Select ChordSequence <2>
Select ChordSequence <1>

Please note that whenever you turn on MIDI Listen either in the Piano area of CP or in any of the controller lists, the control of CP via MIDI is automatically disabled to avoid accidental triggering and confusion.

Vel.: This slider / list sets the override velocity value, which is used in the RhythmProg part of CP in case the per note velocity values are disabled (the {*} button of V-Ctrl. in the RhythmProg editor is enabled).

Channel: Sets the MIDI channel of all data being send by ChordProgressor.

Ctrl. 1-4: The 4 controller sliders to the right side of the Midi-Set area define C1-C4 of the RhythmProg:



Set MIDI Controller for RhythProgCtrlr1:

0:BankSelect MSB	34:BreathCtrl LSB	68:Legato on/off	102:Free Ctrl 23	136:Juno1
1:Modulation MSB	35:Free Ctrl 1 LSB	69:Hold 2 on/off	103:Free Ctrl 24	137:Juno1
2:BreathCtrl MSB	36:Foot Ctrl LSB	70:Sound Ctrl 1 Variation	104:Free Ctrl 25	138:Juno1
3:Free Ctrl 1 MSB	37:Portamento LSB	71:Sound Ctrl 2 Harmonic	105:Free Ctrl 26	139:Juno1
4:Foot Ctrl MSB	38:DataByte RPN-/NRPN-Ctrl LSB	72:Sound Ctrl 3 Release	106:Free Ctrl 27	140:Juno1
5:Portamento MSB	39:MainVolume LSB	73:Sound Ctrl 4 Attack	107:Free Ctrl 28	141:Juno1
6:DataByte RPN-/NRPN-Ctrl MSB	40:Balance LSB	74:Sound Ctrl 5 Brightness	108:Free Ctrl 29	142:Juno1
7:MainVolume MSB	41:Free Ctrl 2 LSB	75:Sound Ctrl 6	109:Free Ctrl 30	143:Juno1
8:Balance MSB	42:Panorama LSB	76:Sound Ctrl 7	110:Free Ctrl 31	144:Juno1
9:Free Ctrl 2 MSB	43:Expression LSB	77:Sound Ctrl 8	111:Free Ctrl 32	145:Juno1
10:Panorama MSB	44:Effect Ctrl 1 LSB	78:Sound Ctrl 9	112:Free Ctrl 33	146:Juno1
11:Expression MSB	45:Effect Ctrl 2 LSB	79:Sound Ctrl 10	113:Free Ctrl 34	147:Juno1
12:Effect Ctrl 1 MSB	46:Free Ctrl 3 LSB	80:General Ctrl 5	114:Free Ctrl 35	148:Juno1
13:Effect Ctrl 2 MSB	47:Free Ctrl 4 LSB	81:General Ctrl 6	115:Free Ctrl 36	149:Juno1
14:Free Ctrl 3 MSB	48:General Ctrl 1 LSB	82:General Ctrl 7	116:Free Ctrl 37	150:Juno1
15:Free Ctrl 4 MSB	49:General Ctrl 2 LSB	83:General Ctrl 8	117:Free Ctrl 38	
16:General Ctrl 1 MSB	50:General Ctrl 3 LSB	84:Portamento Ctrl 2	118:Free Ctrl 39	
17:General Ctrl 2 MSB	51:General Ctrl 4 LSB	85:Free Ctrl 17	119:Free Ctrl 40	
18:General Ctrl 3 MSB	52:Free Ctrl 5 LSB	86:Free Ctrl 18	120:All Sounds off	
19:General Ctrl 4 MSB	53:Free Ctrl 6 LSB	87:Free Ctrl 19	121:All Ctrl Reset	
20:Free Ctrl 5 MSB	54:Free Ctrl 7 LSB	88:Free Ctrl 20	122:Local Ctrl on/off	
21:Free Ctrl 6 MSB	55:Free Ctrl 8 LSB	89:Free Ctrl 21	123:All Notes off	
22:Free Ctrl 7 MSB	56:Free Ctrl 9 LSB	90:Free Ctrl 22	124:Omni off	
23:Free Ctrl 8 MSB	57:Free Ctrl 10 LSB	91:Effect 1 Ext. Level	125:Omni on	
24:Free Ctrl 9 MSB	58:Free Ctrl 11 LSB	92:Effect 2 Tremolo Level	126:Mono on/Poly off	
25:Free Ctrl 10 MSB	59:Free Ctrl 12 LSB	93:Effect 3 Chorus Level	127:Poly on/Mono off	
26:Free Ctrl 11 MSB	60:Free Ctrl 13 LSB	94:Effect 4 Detune Level	128:Polyphonic Aftertouch	
27:Free Ctrl 12 MSB	61:Free Ctrl 14 LSB	95:Effect 5 Phaser Level	129:Monophonic Aftertouch	
28:Free Ctrl 13 MSB	62:Free Ctrl 15 LSB	96:Data Increment RPN/NRPN	130:Pitch Wheel	
29:Free Ctrl 14 MSB	63:Free Ctrl 16 LSB	97:Data Decrement RPN/NRPN	131:Program Change	
30:Free Ctrl 15 MSB	64:Hold 1 on/off	98:NRPN LSB	132:Song Select	
31:Free Ctrl 16 MSB	65:Portamento on/off	99:NRPN MSB	133:Juno106: LFO Rate	
32:Bank select LSB	66:Sustenuo on/off	100:RPN LSB	134:Juno106: LFO Delay	
33:Modulation LSB	67:Soft Pedal on/off	101:RPN MSB	135:Juno106: DCO LFO	

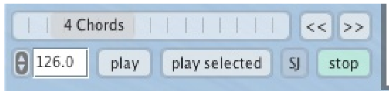
With controller 1 - 4 (C1-C4) defined in the Midi-Set section of CP you are enabled to control up to four modulation destinations in your sound device per RhythmProg step (please see below in the section about RhythmProg for more details). The above shown list is preset with the standart MIDI control change definitions plus some examples for user definable SysEx controllers at the end of the list. The {MIDIListen} function offers you an easy way to determine the controller number of your choice, if your sound device sends controller data on the turn of its knobs. Please note that so far NRPN/RPN multi MIDI message controllers are not supported by CP, but only single message controllers and SysEx controllers.

Please note that due to flaws in the Mac OS X Java 1.6 javax.sound.midi implementation no short SysEx messages are transmitted. The only workaround so far is to use the mmj 0.94 Java MIDI library by N. Peters.

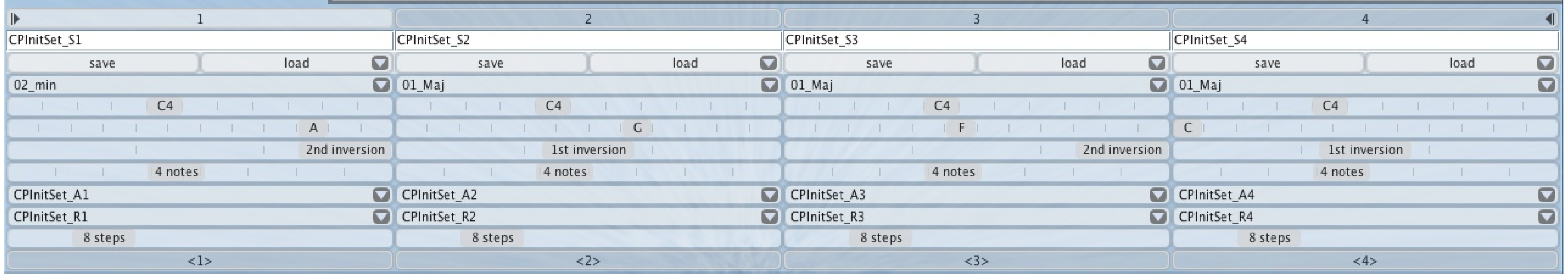
The {MIDISend} button simply sends out the currently selected controller with a randomly varying value.

<nerd hint #3: when CP is launched the very first time, it automatically generates the two files "MIDIControllerList.xml" and "SysExControllerList.xml" which contain all text definitions you see in the above shown list. In case you want to adapt these to your setup or liking, you may edit the strings in these two files. There is one mandatory convention here though: each controller name MUST start with its number immediately followed by ":" then directly followed by any text of your choice (no returns/new lines/control characters though). Please don't mess with the xml tags like <entry>, etc. in these files - if you spoil these, CP might stop behaving properly. Adapting "SysExControllerList.xml" is a bit more complicated, please refer to the documentation comment in the header of the file and the MIDI SysEx documentation of your sound device.>

[3.4] ChordProgressors ChordSequence (ChordSeq) section



Ok, now we are getting really into it: The upper middle of ChordProgressor gives you all the control over the used chords, arpeggios and rhythms per CP-Set:



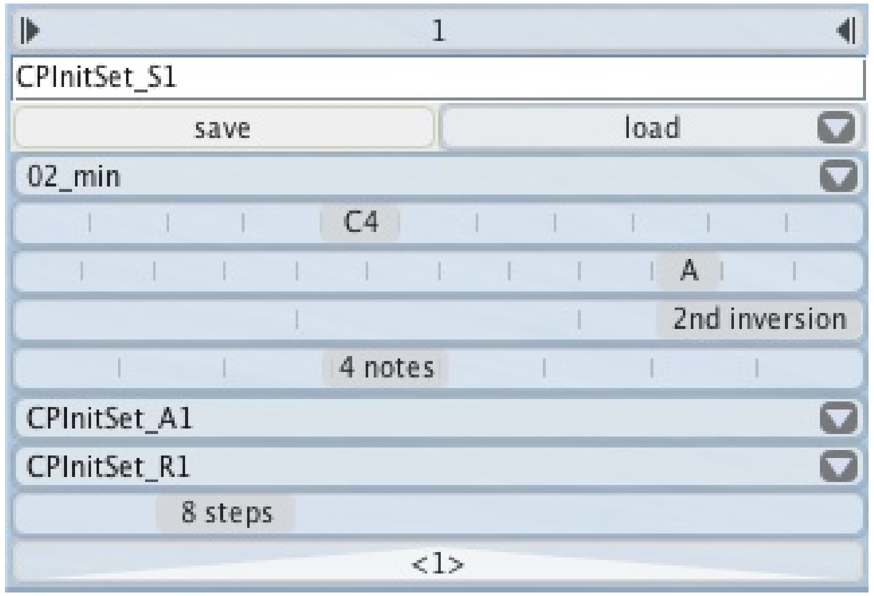
Chords: The chord(s) slider lets you select how many chords you'd like to play in the total sequence / loop. You can define up to 16 wildly different chords here. Note that even if you lower the chord number (temporarily), CP will keep your hidden chords at the end in his memory.
<<;>>: Use these buttons to rotate the ChordProgression inside the LoopLocators which you can set with SHIFT-mouse-click (left) and CTRL-mouse-click (right) or right-mouse-drag-and-drop-click.

DragButton & Textfield Tempo: On the drag button click and hold, then drag the mouse up or down to continously change the tempo. The text field next to it will update to the altered tempo. Alternately you may change the tempo by entering it precisely in the text field. As said above, please be advised to change the value to the newly entered by either pressing Enter/Return or the Tab key. If CP is in MIDI Clock slave mode, the tempo text field shows the tempo resolved from the incoming clock. This value may vary some from the one set in the MIDI Clock master due to measurement and system clock differences.

play: play the whole CP sequence in a loop until you can't hear it anymore
play selected: just play the currently selected ChordSeq step in an everlasting loop, e.g. for tweaking the ArpProg and RhythmProg
stop: ok, now you've got enough of it - then press this button

Per ChordProg step you've got a ChordSeq with the following controllers to set the shape of things:
On top there is the TimeRuler / PlayIndicator which shows the currently playing ChordSeq which can be used to select the next to be played ChordSeq as well. If not in Edit-Mode (which is described in detail further below) the clicking on the TimeBar also selects the shown Chord, ArpProg and RhythmProg of the ChordSeq in the edit area. In Edit mode you use the lowermost EditSelectionRuler to select the ChordSeq(s) of which the Chord, ArpProg and RhythmProg parts can be edited.

If you click there while holding down the Shift-key and drag it over the TimeBar, you'll see the cursor changing from ◁ ▷ to ◁–▷ and you can drag the first clicked ChordSeq to another position, where it is inserted and shift all other CordSeq from there to the right.
If you do the same with additionally holding down the ALT-key, the cursor will change to ◁△▷, indicating the copy/overwrite modus, which copies the first clicked ChordSeq and overwrites the ChordSeq where you release the mouse button. With Shift-CMD/WIN-key you'll get the multiple copy mode ◁⩓▷ which will overwrite all ChordSeq with the 1st clicked one from where you drop it.



the TimeRuler

the name of the ChordSeq

save or load ChordSeq from library

select a Chord preset

base octave of the ChordSeq

the root key of the chord

the inversion of the chord

number of used notes (1 - 8)

select an ArpProg preset

select an RhythmProg preset

number of used RhythmProg steps

the EditSelectionRuler

The EditSelectionRuler has also the control over which ChordSeq(s) Chord, ArpProg and RhythmProg is edited in the edit section below it, if the EditMode is enabled (- edit):
- a click on the EditSelectionRuler selects the ChordSeq above it for editing
- a center mouse button click toggles on/off additional ChordSeqs for MultiEdit
- a right mouse click, drag and drop selects all ChordSeqs between the 1st clicked and the one where the mouse button is released for MultiEdit
- ALT-key plus mouse click selects all ChordSeqs for MultiEdit

In MultiEdit for all the above described selection options applies, that the first clicked ChordSeq is the prototype, whichs settings are copied over to the other selected ChordSeq as soon as you change something in the ArpProg or RhythmProg editor, so please use it consciously. Please note that while CP is in MultiEdit by having more than one ChordSeq selected in the EditSelectionRuler you can't apply changes in the Chord editor, since this would wipe out your carefully choosen ChordProgression in the above section of CP.

Some more words about the concept of the ChordSeq in ChordProgressor:

A ChordSeq contains all the definitions regarding a single chord of which you may have up to 16 in a ChordProgressor sequence and which can be stored to and loaded from the CP-Library in addition of being a part of the current CP-Set.

The Chord preset selected in the 4th line of the ChordSeq section contains the information which keys / notes are used for the chord to be played normalized to C.

This chord is then shifted by the setting of the base octave and root key to the desired chord. If the {inversion} slider is set to anything but "root", the chord is played in the selected inversion. This resulting chord after all the above described operations may then be used with a note count starting with the lowest note of the chord only up to eight notes in total.

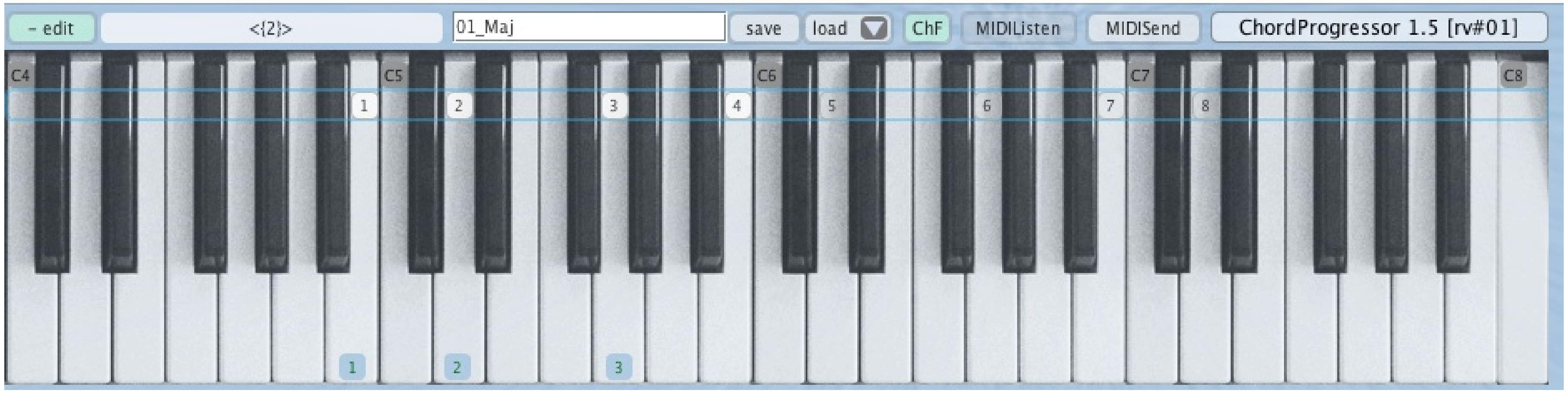
From these chord notes the selected ArpProg chooses the to be played notes controlled in their time position, length, velocity and modulation by the selected RhythmProg. The ArpProg only plays active note steps of the RhythmProg. If the number of notes set in the ChordSeq is below the number of defined notes in the ArpProg the ArpProg folds the otherwise unplayable notes down to the notes enabled in the ChordSeq. If you play with the {note(s)} and {step(s)} sliders of a ChordSeq selected for editing, you'll easily recognize the dependencies due to ChordProgressors shading of disabled parts in the ArpProg and RhythmProg editors.

The number of used steps from the RhythmProg is detemined by the {step(s)} sliders setting of the ChordSeq.

If either the length of the ArpProg or the RhythmProg pattern is smaller then the {step(s)} setting of the ChordSeq, they'll get repeated the neccessary number of times to fill up the step number of the ChordSeq. This design concept enables you to easily archieve complex changing modulation patterns by simply setting odd / different sequence lengths in ChordSeq, ArpProg and RhythmProg. At the end of the maximum number of 36 steps per ChordSeq / RhythmProg the pattern gets always reset to its inital state though.

[3.5] ChordProgressors Chord section

The left center of ChordProgressor shows a piano keyboard to set and display the playing / selected chord:



Leftmost you'll find the {+/- edit} Button which enables / disables the Edit-Mode of the complete lower part of CP. If disabled and {animation} is enabled the lower part of CP follows the played part of the chord progression with animating the played keys. The text display right next to the {edit} button shows the list of the selected ChordSeqs. In the above example ChordSeq#1 and ChordSeq#2 are selected for editing and #1 (the one in {}-brackets) is the prototype from which the settings are copied over to #2.

Next to the right you'll find the usual name text field, a {save} and a {load} button, which work accordingly to their counterparts in the ChordSeq section. Please note that you can't overwrite the predefined presets of CP, so please chose different names when saving a new chord which are not already part of the CP-Library. <nerd hint #4: you can create your own 'write protected' presets by using a name of the scheme NM_thisIsMyChordsName where NM is a two digit number.>

{ChF} is the abbreviation for ChordFinder. If enabled the entered keys/notes are compared to already defined chords in CP's library and if there is a matching chord found it will get automatically selected. You may use this function to find out which type of chord it is you've been accidentally playing. Please note that if you don't stick to the common chords, you may be better of disabling {ChF} to avoid it 'correcting' your played chords to the prototypic ones.

If the Editor-Section is enabled and not in MultiEdit mode you can edit the keys / notes of the currently selected Chord preset here by clicking on the keyboard to toggle keys in the chord on/off. Or you enable {MIDIListen} to enter your desired chord keys via your MIDI keyboard. The way of entering chords this way works like this: You can press and hold either one key or the complete chord, as long as you keep holding down at least one key, all additionally pressed keys are added to the chord until you have released the last key. To change the chord you can either toggle singles notes via mouse click or start a new one by pressing keys on your MIDI keyboard.

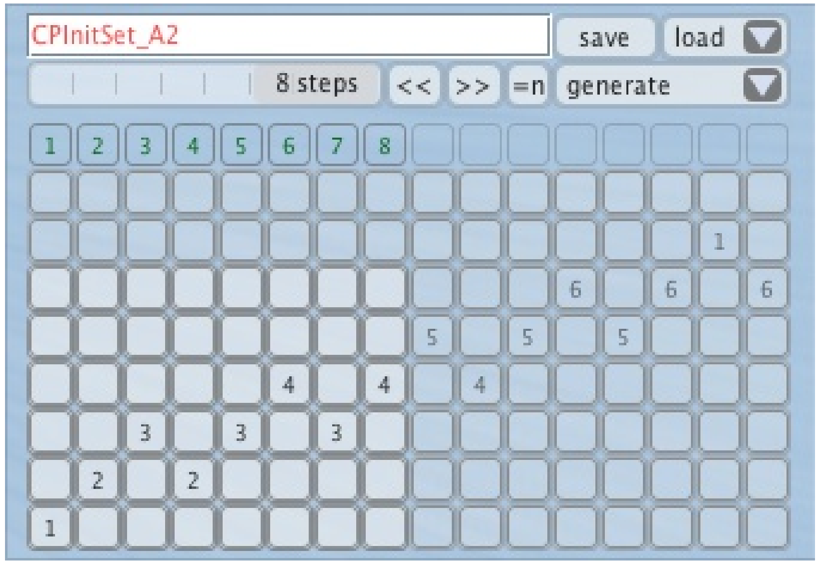
Doing this while CP is playing in the "single play" mode, you have a bit of a classical arpeggiator, but with the extended functionality of the ArpProg grid and the RhythmProg. As CP's concept is more intended for the creation of loops than for life performance, please be advised, that CP is not the optimal choice for reliable rock steady life playing. Despite this being said, you may use it in these fields with a powerful computer at your own risk.

The {MIDISend} button simply sends out the current chord with randomly varying velocity values.

The labels on the piano keyboard have the following meaning:
Top row: these labels show the shifted position of CP's 'virtual' piano keyboard on a fullscale 'real' piano.
Row below: this row shows all the keys contained in the selected chord and the displayed numbers relate to the key numbers shown in the ArpProg grid to the right of CP's piano keyboard.
Bottom row & up: these labels show the generated notes by the arpeggiator numbered by their sequence of playing.

[3.6] ChordProgressors ArpProg section

The right center of ChordProgressor shows the ArpProg grid to program any kind of complex arpeggio:



The topmost row contains the usual name text field as well as a {save} and {load} button and may be used to add or recall arpeggios from the CP-Library.

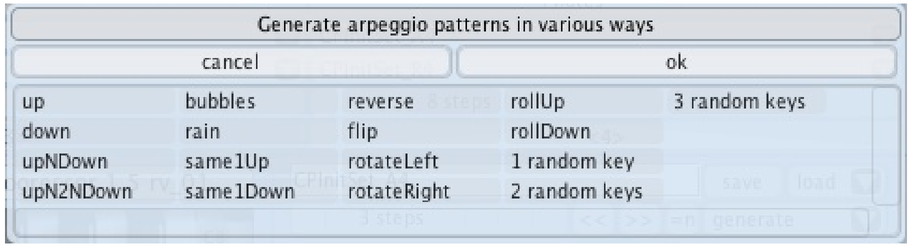
The {steps} slider at the start of the next row sets the number of used steps from this arpeggio pattern. CP's ArpProg has a maximum of 8 notes per step over the course of a maximum of 16 steps.

Each grid cell which shows a number plays the key with that number from the chord to the left (shown there in the 2nd row).

Played keys can be toggled on/off by left mouse click, clicking with the right mouse button resets the whole column to the one key you've just clicked on. Please note that each ArpProg step has to have at least one key enabled as pauses are defined in the RhythmProg.

The {<<} and {>>} buttons shift / rotate the active part of the arpeggio pattern by one key to the left or the right.

The {generate} button opens up a list with various functions to modify or generate arpeggio patterns:



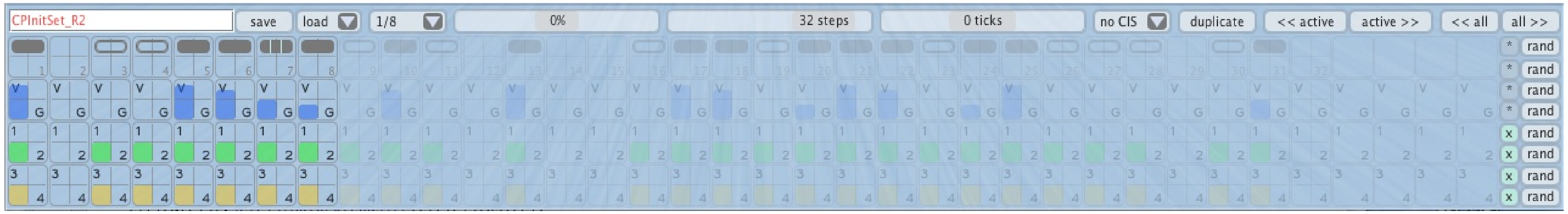
up:	generates a classical up arpeggio (*)
down:	generates a classical down arpeggio (*)
upNDown:	generates a classical up & down arpeggio (*)
upN2NDown:	generates a up & down arpeggio with two same notes at the top & bottom (*)

bubbles:	generates a bubbling up arpeggio (*)
rain:	generates a drippling down arpeggio (*)
reverse:	this function reverses the active part of the arpeggio
flip:	flips the keys of the active part of the arpeggio upside down
rotateLeft:	this function duplicates the {<<} button described above
rotateRight:	this function duplicates the {>>} button described above
rollUp:	rolls / rotates the active part of the arpeggio by one key up
rollDown:	rolls / rotates the active part of the arpeggio by one key down
1 - 7 random key(s):	generates a random key pattern with 1 to 7 keys per step. Please note that the number of available keys for these list entries depends on the number of resulting chord keys set by the used notes controller in the ChordSeq section. (*)

(*): this function works always on the whole arp grid and adjusts the arpeggio step length to the one set in the ChordSeq

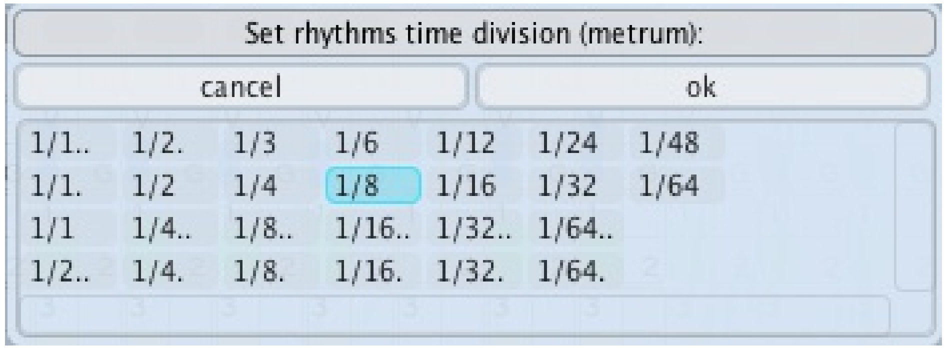
[3.7] ChordProgressors RhythmProg section

The complete lowermost part of ChordProgressor is covered with the RhythmProg section:



As usual the first control elements in this section of CP are to save and load RhythmProg patterns to the CP-Library.

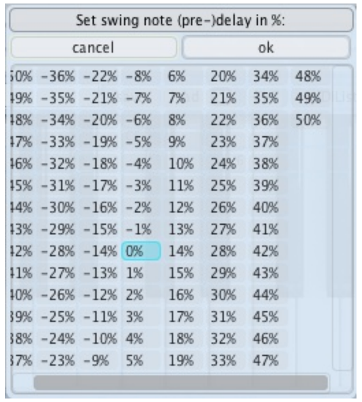
The next list control to the right offers common options regarding the beat subdivision relative to the chosen tempo in the ChordSeq section of CP:


Set rhythms time division (metrum):

cancel | ok

1/1..	1/2.	1/3	1/6	1/12	1/24	1/48
1/1.	1/2	1/4	1/8	1/16	1/32	1/64
1/1	1/4..	1/8..	1/16..	1/32..	1/64..	
1/2..	1/4.	1/8.	1/16.	1/32.	1/64.	

The percent slider to the right sets the swing (pre-)delay in percent of the beat subdivision. E.g. +50% at 1/8 means 1/8+1/16; -25% at 1/4 means 1/4-1/16, etc.

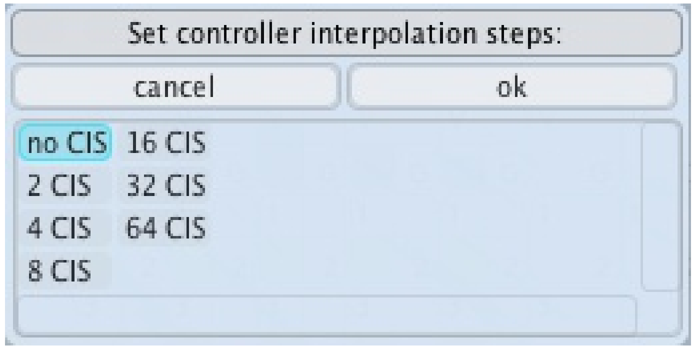

Set swing note (pre-)delay in %:

cancel | ok

50%	-36%	-22%	-8%	6%	20%	34%	48%
49%	-35%	-21%	-7%	7%	21%	35%	49%
48%	-34%	-20%	-6%	8%	22%	36%	50%
47%	-33%	-19%	-5%	9%	23%	37%	
46%	-32%	-18%	-4%	10%	24%	38%	
45%	-31%	-17%	-3%	11%	25%	39%	
44%	-30%	-16%	-2%	12%	26%	40%	
43%	-29%	-15%	-1%	13%	27%	41%	
42%	-28%	-14%	0%	14%	28%	42%	
41%	-27%	-13%	1%	15%	29%	43%	
40%	-26%	-12%	2%	16%	30%	44%	
39%	-25%	-11%	3%	17%	31%	45%	
38%	-24%	-10%	4%	18%	32%	46%	
37%	-23%	-9%	5%	19%	33%	47%	

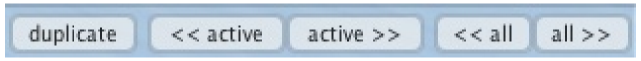
The {steps} slider sets the total number of used RhythmProg steps.

The {ticks} slider sets the +/- offset between the notes and the corresponding controllers 1-4 in ticks (1/384) to give a means to compensate delayed modulation processing by the connected sound device or as a musical effect.

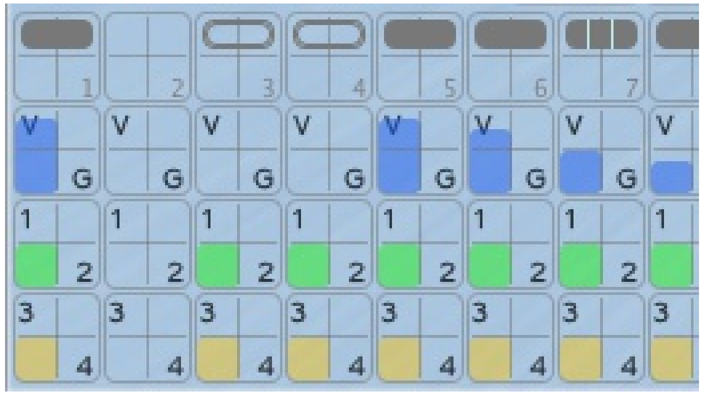


The Controller Interpolation Steps (CIS) function may be activated if you prefer smooth modulation controller rides between the notes instead of jumping controller values at the note start respectively the note start plus the {ticks} sliders (pre-)delay value. The number before CIS indicates the number of subdivisions inbetween two note steps of the ChordProg.

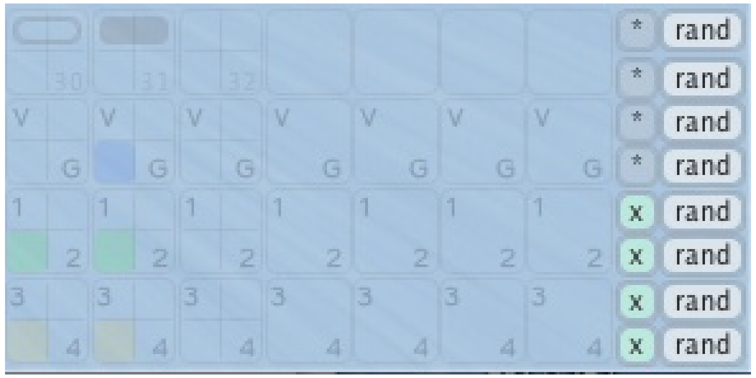
The {duplicate} button duplicates the active part of the RhythmProg after this part. If pressed with the right mouse button, multiple duplicates will be copied over the complete rest of the RhythmProg up to the step number set in the RhythmProg.



With the {<< active} and {active >>} buttons you can shift / rotate the active part of the RhythmProg by one step at a time. Using the {<< all} and {all >>} buttons the shift / rotate operation takes place on the complete RhythmProg defined by its number of steps.
The grid of the RhythmProg consists of four parts per step, with which you can control if a note is played or only its controller data, when the note is played relative to the grid, the notes velocity and gate time (length) and up to 4 different controllers per note. A left click in the upper part of the top row toggles a note and its controllers on/off, whereas a right click toggles the controller only mode on/off. A center mouse click divides a note in up to 8 flam/roll notes, shift center click reduces the number of (sub-)notes.



The next row from the top controls the velocity (V) and gate (G) time of each note, with the half of the controller field meaning a velocity of 64 and a gate time of excactly the rhythms time division beat length. So if you'd like to use the glide of your synth, the gate controller should be slightly right of the middle. The controllers (1)-(4) will send MIDI Control Changes according to the settings made in the Midi-Set section of CP.

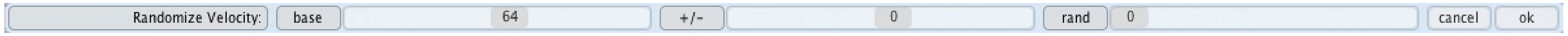


On the very right side of each row are {*} and {rand} buttons with the following functions:

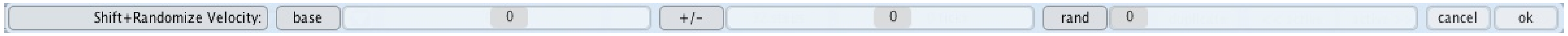
The {*} buttons override / disable all individual per note control values with (from top to bottom):
1) note on, 2) no note shift, 3) velocity set in the Midi-Set section, 4) gate = time division beat length, 5)-8) no controller data will be sent / generated if activated.

The {rand} button of the top most row randomizes the note on/off switch.

All other {rand} buttons have two different functions depending on wheter been pressed with a left or a right mouse button click. A left click activates the absolute mode controls:



Whereas a right click calls up the relative mode controls:



In the absolute mode the resulting values are calculated by {base}+/-{rand}, in the relative mode the values are the result of {old value from the grid}+{base}+/-{rand}.

[4] Quirks and issues

Besides its carefully developed feature set, Chordprogressor has two major quirks by design:
a) there is no undo (except for a few dialogs, which restore the previous state if left with cancel)
b) the realtime performance most likely can't rival the one of your favourite DAW.

Why?

a) An undo function for an application or plugin which has a rich set of realtime parameters which can be changed intuitively by the user is likely to generate substancial data, which has to be stored somewhere. To be able to use ChordPorgressor on as many platforms as possible while at the same time keeping the development time in a reasonable bandwidth, JAVA was chosen as the development platform. While JAVA takes care of all the memory allocation handling (which eases the development process significantly), this at the same time means, you don't have much control over when JAVA handles the memory management. But when it does, it might seriously interfere with other processes going on. Means: the timing goes down the drain. So one development aim was to keep the memory usage / fluctuations as low as possible. Which is diametrical to a deep undo functionality. Additionally the implementation of an consistent undo is anything but trivial and will therefor eat up substancial development time. So ChordProgressor is functioning like real life: no undo.

b) As detailed above JAVA has been chosen as platform, which is indeed a bit adventurous when it comes to realtime requirements. So some of the initial concerns proved to be right, but in the end a workable solution was found. There seems to be another possible option for the sequencing engine in CP in JAVA, but that would have meant that CP wouldn't work on older operating systems without javax realtime support. The result of this tradeoff is, that CP's timing is mostly decent, with a few hickups here and there depending on how much pressure you put on your system, while running CP.

So if you consider either a) or b) a show stopper, CP might not be the right choice for you.

[4.1] Mac OS X issues

As ChordProgressor uses the Java engine delivered with Mac OS X (on the latest Mac OS X versions you might need to install the Apple Java build via Softwareupdate or the AppStore) it is dependent on Apples implementation flavour. Unfortunately this means the current Java version (Java 6) is neither up to date nor completely flawless.

Known issues are:
- Quiting with CMD-Q aborts all without saving anything (please use the {quit} button instead, except you want to leave the program without saving preferences etc.)
- It is not possible to send short SysEx messages with the current Apple Java implementation. The only workaround for this is installing the mmj Java extension <LINK>, though this will result in double entries in the MIDI input and MIDI output selection lists.

[4.2] Windows issues

As Windows is the operating system with one of the most diverse system constellations it is practically impossible for a small company to test a given software on even most systems. That said, you should definitely test the demo version of ChordProgressor thourougly on your machine, before using the regular version seriously.

Known issues are:
- Depending on your system configuration the timing precision of CP may get corrupted by other tasks running in parallel and gaining higher priority.
- The timing right after starting to play tends to be degraded on slower systems.

[4.3] Linux / Unix issues

ChordProgressor has been tested on Mint Linux only, so since there are a lot of different Linux and other Unix like distributions you definitely should seriously test the CP demo version on the operation system of your choice. As the Linux variant of CP is a slim package as it is for Mac OS X, you need to take care of the Java installation yourself via the package manager of your operating system.

Known issues are:
- The closing of a MIDI input port does not seem to work completely, so on the 2nd start of CP there is no midi input available. Unfortunately this requires a system restart to be fixed until the next closing of a MIDI input port.
- When there is MIDI clock input data coming in while trying to open the corresponding MIDI input, the request and setup of that MIDI inout port is likely to fail.
- Since CP creates a library folder next to the CP application, it requires write and modification access rights to that folder. In case of lacking these no data can be saved. To resolve this issue create a folder named "ChordProgressorData" with sufficient access priviliges.

version of 15.08.2014