CLAVIER-36 User Manual

Version 0.0.0

OVERVIEW

CLAVIER-36 (alternatively C36) is a programming environment for generative music. Programs are laid out in a two-dimensional grid, and evolve over time according to a fixed set of rules. The system is much like a cellular automaton, in that most of the rules governing the evolution of the system are local.

C36 programs describe sequences of discrete events in time. The environment includes a primitive sampler, as a self-contained means of interpreting these events as sound. For full expressivity, though, the system is best used as a generator of data for interpretation by an external musical instrument, such as a synthesizer. As of the time of writing, the only supported communication protocol is MIDI. In the future, we may support other protocols.

Distribution

The software is distributed through Steam for Windows and macOS. There is also a limited version that runs in the browser. The limitations of the browser version are as follows:

1. MIDI output is not implemented.
2. The size of loaded samples is limited to 2MB.
3. Performance is not quite as good as the native version.

Keyboard Shortcuts

Throughout this manual, some of the available keyboard shortcuts are listed. If you are on macOS, the Command key functions in place of the Control key, and the Option key functions in place of the Alt key.

Fair Warning

C36 is under active development. The semantics of the language will change significantly. Patches created prior to the release of version 0.1.0 will likely not load correctly in future versions of the software.

Acknowledgement

C36 would not have been possible without inspiration from Orca. Many thanks to Devine and Rekka for creating such a lovely piece of software.

THE EDITOR PANEL

The editor panel is the main area in the center of the window. It shows the state of the currently running program.

Editing the Selection

- Select a region by dragging between cells with the left mouse button.
- Move the selection by pressing the arrow keys.
- Grow the selection by holding Shift and pressing the down/right arrow keys. Shrink the selection by holding Shift and pressing the up/left arrow keys.

Moving the Camera

- Pan the camera by dragging with the right mouse button, or by holding Alt and dragging with the left mouse button.
- Center the camera on the current selection with Control-M.
- Zoom the camera by hovering the mouse over the editor panel and moving the mouse wheel.

Editing the Program Text

- Place literals by pressing the corresponding alphanumeric key.
- Place operators by holding Shift and pressing the corresponding key (see the reference card in the right panel, or the reference section of this manual).
- Delete by pressing Backspace or Delete.
- Cut with Control-X, copy with Control-C, and paste with Control-V.
- Increment selected literals by pressing Control-Up. Decrement selected literals by pressing Control-Down.
- Toggle the power state of selected operators by pressing Enter.

Dataflow Wiring

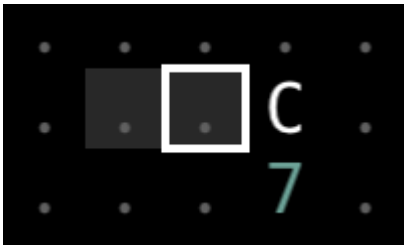
- Create dataflow wires by holding Control and dragging with the left mouse button between two cells.
- Delete dataflow wires by selecting either endpoint and pressing Backspace. Alternatively, draw an identical wire to an existing wire to delete it.

WALKTHROUGH

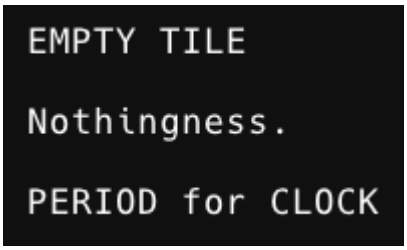
In this section, we will walk through the creation of a small C36 program, to get a feel for the language.

Start by placing a clock. This is done by holding Shift and pressing the C key. Once placed, the clock will write its output to the south.

Notice that the two cells immediately to the west of the clock have been highlighted. These are its inputs. Move the cursor to the rightmost input cell.

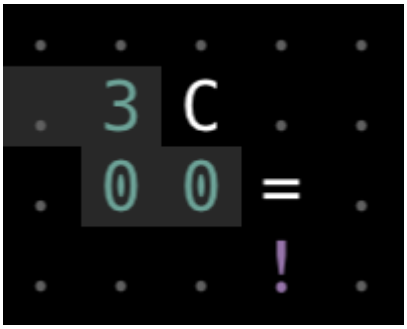


The information panel to the right describes the selected cell. The first two lines describe the contents of the cell. The remaining lines list any operations the cell is currently an input for. In this case, we are being told that this cell is being used as the period for the clock. The period is the number of cycles the clock will count before repeating.

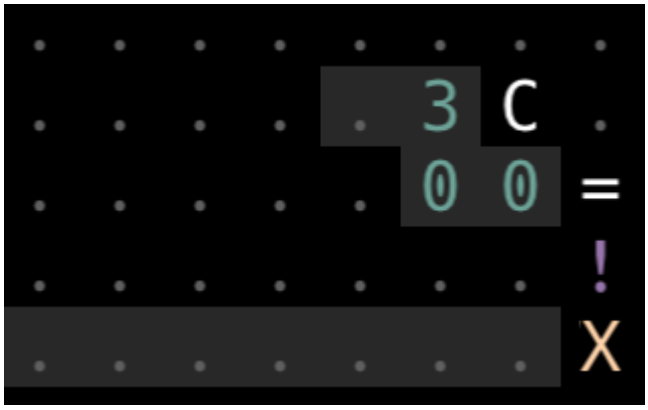


Let's set the period to 3. Observe the new behavior of the clock.

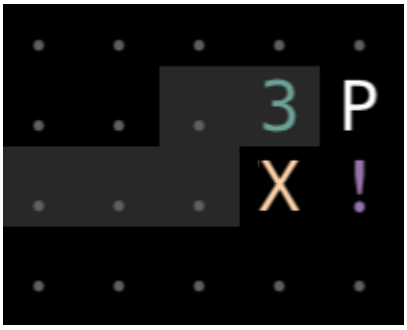
We can use the periodicity of the clock to create a regular pulse, which we'll use to trigger sample playback. Feed the output of the clock into the right hand side of an equality, and set the left hand side to zero. The equality will produce a bang whenever both inputs are equal.



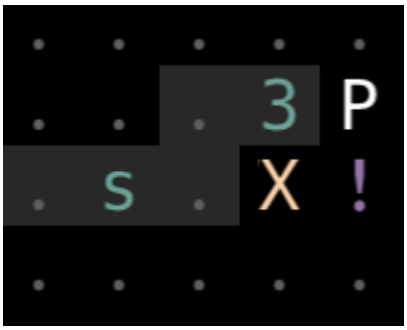
At this point, we're ready to play some sound. Place a sampler directly underneath the output of the comparator. You should hear a repeating tone. The sampler operator will start playback of a new voice each cycle a bang is present in any of the four adjacent cells.



This construction involving a clock and a comparator producing a regular bang is so useful that C36 includes a dedicated operator for it: the "pendulum". (Now is perhaps a good time to ask for forgiveness for some of the operator names.) We can replace the clock and the comparator to save space as follows.

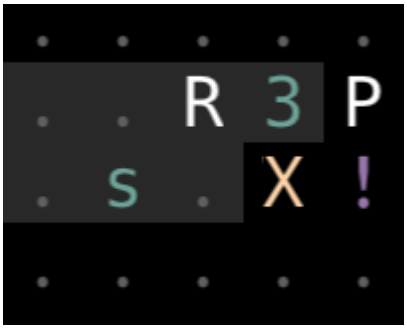


The sampler takes seven inputs specifying the character of the sound to play. Three of them (attack, hold, and release) constitute what is known as a volume envelope. Try setting the release to `s` to extend the length of each note.

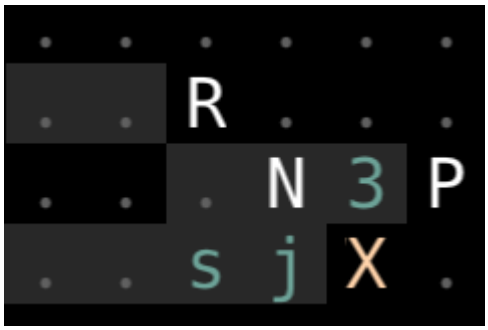


The numeral system of C36 is base 36: a happy coincidence of three octaves of range with the number of alphanumeric keys on an English keyboard. The letter `s` is the way we write `28`.

Let's try varying the pitch of the samples by feeding randomness into the pitch parameter.



Perhaps we want something a little less atonal. Instead of feeding the randomness directly into the pitch parameter, which is measured in semitones, we can run it through a quantizer.



Let's finish off the composition by adding another voice. Box select the whole construction, press Control-C to copy, move the selection down a few cells with the down arrow key, and press Control-V to paste.

This concludes the walkthrough. See the operator reference section for a full listing of the available operators.

USER INTERFACE

Sampler Panel

Click on any of the available slots to load a wave file from disk. Control-click a slot to unload the sample. Scroll by hovering the mouse above the sample bank and moving the mouse wheel.

Bottom Bar

The following information is displayed in the bottom bar:

1. patch size
2. cursor position
3. selection size
4. tempo
5. cycle count

All of these are editable by clicking on them, other than the cycle count.

Effects Panel

All internal audio is routed through a global reverb. Press F9 or click the item in the View menu to change the reverb settings.

Saving and Sharing Patches

See the File menu to save and load from disk. In the browser-based distribution, there is also the option to generate a shareable link.

EVALUATION ORDER

Each cycle, the grid is processed in English reading order: top to bottom, with each row processed left-to-right. When there is a race between two cells writing to the same cell, the cell that acts later wins.

Dataflow wires are constrained to follow the evaluation order. This has the desirable consequence that it is impossible to form cycles.

There are two ways of moving data from later in the evaluation order to earlier:

- The `I` (INTERFERE) operator, that can create upwards-moving data via the `MOMENTUM` parameter.
- Registers, using the `S` (STORE) and `L` (LOAD) operators. If you write to a register below the place you read it, the written data will not be available until the following cycle.

OPERATOR REFERENCE

0. ADD

`<LEFT ADDEND> <RIGHT ADDEND> +`

Performs modular addition. Empty inputs are treated as zero.

1. SUBTRACT

`<MINUEND> <SUBTRAHEND> -`

Performs modular subtraction. Empty inputs are treated as zero.

2. MULTIPLY

`<MULTIPLIER> <MULTIPLICAND> *`

Performs modular multiplication. Empty inputs are treated as one.

3. DIVIDE

`<DIVIDEND> <DIVISOR> /`

Performs integer division, rounding down. Division by zero is defined to be the constant function returning zero. An empty dividend is treated as zero. An empty divisor is treated as one.

4. MODULO

`<DIVIDEND> <DIVISOR> %`

Computes the remainder. When the divisor is zero, this is defined to be the identity function. An empty dividend is treated as zero. An empty divisor is treated as one.

5. EQUATE

`<LEFT COMPARATE> <RIGHT COMPARATE> =`

Produces a bang when the inputs are equal. Inputs must have the same type to be considered equal. Empty cells are considered to be equal.

6. MORE

`<LEFT COMPARATE> <RIGHT COMPARATE> >`

Produces a bang when the left input is greater than the right input. Inputs must have the same type for the comparison to succeed.

7. LESS

`<LEFT COMPARATE> <RIGHT COMPARATE> <`

Produces a bang when the left input is lesser than the right input. Inputs must have the same type for the comparison to succeed.

8. AND

`<LEFT CONJUNCT> <RIGHT CONJUNCT> &`

When both inputs are literals, performs bitwise AND. Otherwise, produces a bang whenever both inputs are bangs.

9. OR

`<LEFT COMPARATE> <RIGHT COMPARATE> |`

When both inputs are literals, performs bitwise OR. Otherwise, produces a bang whenever either input is a bang.

A. INTERPOLATE

`<TIME> <MINIMUM> <MAXIMUM> A`

Performs linear interpolation.

B. BOTTOM

`<LEFT COMPARATE> <RIGHT COMPARATE> B`

Selects the lesser of the two inputs. Undefined when both inputs are empty.

C. CLOCK

`<INTERVAL> <PERIOD> C`

Marks time. Increments by one every `INTERVAL` cycles, returning to zero after `PERIOD` increments.

Clocks can be offset by manually setting their output. Clocks are the only operators that make use of feedback in this way.

I. INTERFERE

`<SYMBOL> <DIRECTION> <X> <Y> I`

Writes `SYMBOL` to program memory at the relative coordinate specified by `X` and `Y`. `X` grows to the east, and `Y` grows to the south. The written symbol will move autonomously according to the following table, at a rate of one cell per cycle:

`DIRECTION`	Direction
Empty	Stationary
0	North
1	East
2	South
3	West

`DIRECTION` is interpreted mod 4. When the written symbol collides with another symbol or the edge of proram memory, it is destroyed.

L. LOAD

`<REGISTER> L`

Loads the symbol stored in the specified register.

M. MULTIPLEX

`<X> <Y> M`

Reads the symbol stored in program memory at the relative coordinate specified by `X` and `Y`. `X` grows to the east, and `Y` grows to the north.

N. QUANTIZE

`<INDEX> N`

Computes the semitone value of note number `INDEX` of the major scale.

P. PENDULUM

`<MULTIPLIER> <MULTIPLICAND> P`

Produces a cyclic bang with period equal to the modular product of the inputs. Empty inputs are treated as one.

Q. QUOTE

`<INDEX> Q`

Converts literals to operators.

R. RANDOM

`<INTERVAL> <DIVISOR> R`

Chooses a random literal between zero and `DIVISOR` every `INTERVAL` cycles.

S. STORE

`<SYMBOL> <REGISTER> S`

Stores the symbol in the specified register. Load it elsewhere with the load operator.

T. TOP

`<LEFT COMPARATE> <RIGHT COMPARATE> B`

Selects the greater of the two inputs. Undefined when both inputs are empty.

U. UNQUOTE

`<INDEX> Q`

Converts operators to literals.

W. MIDI CC

`<DEVICE> <CHANNEL> <CONTROLLER> <AMOUNT> W`

Sends MIDI CC messages.

Input	Meaning
`DEVICE`	Which connected MIDI device to transmit to.
`CHANNEL`	Which channel to send the message on. Channel numbers are zero-indexed. Messages on channels above 15 will be ignored.
`CONTROLLER`	Which controller to modulate.
`AMOUNT`	The value to set the specified controller to.

X. SAMPLE

`<SAMPLE> <OFFSET> <VOLUME> <ATTACK> <HOLD> <RELEASE> <PITCH> X`

Whenever a bang is present in an adjacent cell, creates a new sampler voice. The total number of active voices is limited to 512 (if you need more, please let me know).

Input	Meaning
`SAMPLE`	Which sample to play.
`OFFSET`	Where in the sample to begin playback. The sample is divided into 36 equal-length segments.
`VOLUME`	The volume for this voice.
`ATTACK`	The amount of time it takes the voice to fade in.
`HOLD`	The amount of time the voice remains at full volume.
`RELEASE`	The amount of time it takes the voice to fade out.
`PITCH`	The relative pitch of the sample, measured in semitones, centered around `i = 18`.

Y. SYNTHESIZE

`<VOLUME> <ATTACK> <HOLD> <RELEASE> <OCTAVE> <PITCH> Y`

Whenever a bang is present in an adjacent cell, creates a new synthesizer voice. The total number of active voices is limited to 512.

Input	Meaning
`VOLUME`	The volume for this voice.
`ATTACK`	The amount of time it takes the voice to fade in.
`HOLD`	The amount of time the voice remains at full volume.
`RELEASE`	The amount of time it takes the voice to fade out.
`OCTAVE`	The octave to measure pitch from, starting at C0.
`PITCH`	The pitch relative to C, growing upwards, measured in semitones.

Z. MIDI NOTE

`<DEVICE> <CHANNEL> <VELOCITY> <HOLD> <OCTAVE> <PITCH> Z`

Whenever a bang is present in an adjacent cell, sends a MIDI note on message. A corresponding note off will be sent some time later, determined by the `HOLD` parameter.

Input	Meaning
`DEVICE`	Which connected MIDI device to transmit to.
`CHANNEL`	Which channel to send the message on. Channel numbers are zero-indexed. Messages on channels above 15 will be ignored.
`VELOCITY`	The velocity field of the message.
`HOLD`	The amount of time to wait before releasing the note, measured in sixteenth notes at the current tempo.
`OCTAVE`	The octave to measure pitch from, starting at C0.
`PITCH`	The pitch relative to C, growing upwards, measured in semitones.