Version 2 CACE4 0.97.x Rocky River, NSW AUS

Utrecht/Bunnik, NL

6 July 2022

A CASE FOR CACE4?

An introduction to CACE4 (#1)

ABSTRACT

An introduction of the Computer Aided Composition Environment #4.

A MacOSX 64-bit application, for using statistics, A.I. and Information Retrieval techniques for creating Music.

Michèl Koenders, ing., MMus. PhD.

University of the Arts Utrecht. Music & Technology.

Data Analysis by using Statistics, Machine Learning and AI for composing Music.

INDEX.

TABLE OF FIGURES

TABLE OF TABLES

A CASE FOR CACE4?

Author: Michèl Koenders, ing., MMus. PhD.
version 2

"what I cannot create, I do not understand." - Richard Feynman.

PREFACE.

An introduction of the Computer Aided Composition Environment #4[1]. A MacOSX[2] application, for using statistics and Information Retrieval (IR) and Machine Learning (ML) techniques on data-sets initially unrelated to Music. These IR techniques can either be 'simple' statistical tools like: calculating the mean, median or correlation of a data-set. But can also be more elaborated data analysing techniques as there are Hierarchical Cluster Techniques (HCT): *k*-means(++) or Expectation-Maximisation (EM), and Machine Learning (ML) techniques as linear prediction and Artificial Neural Networks in this case the Adaptive Resonance Theory (ART2) can be applied as well. All these techniques are presented to the user of the CACE4 program as simple boxes, with a simple Graphical User Interface (GUI). They can be linked (chained) together in a CACE4 Project window to deploy more elaborated strategies, adaptive to specific needs defined by the user. All these data processing techniques are used for creating note-based output which for now, can be saved as a Standard MIDI Files (SMF, type 0)[3].

[1] Version 0b.97.21 of CACE4: July 2022.

[2] The choice for MacOSX is obtained by the license for this LISP Integrated Development Environment (IDE) as offered by LispWorks.com. For the development of CACE4 LispWorks has been chosen as the preferred LISP Integrated Development Environment (IDE). Its extended set of Libraries (Libs) and Graphical User Interface (GUI Libs as CAPI) made this application possible as it is. See the URL of LispWorks: http://www.lispworks.com for further details about this LISP version. CACE4 is completely written in the well know dialect of Common Lisp according ANSI standard.

[3] Standard MIDI Files (SMF for short) is a specific file format for storing MIDI data organised in tracks according to the specifications defined by the MIDI association. URL: https://www.midi.org/specifications/item/standard-midi-files-smf SMF type 0 is a single track with all of the MIDI data (as program changes and tempo markers). SMF type 1 (multiple tracks with independent tempo tracks) and Music eXtensible Markup Language (MXML) will be implemented in the Autumn/Winter of 2022. A first preliminary version of MXML has been implemented in 0b.97.21, but needs much more attention.

DO WE NEED ANOTHER ALGORITHMIC COMPOSITION ENVIRONMENT/PROGRAM?

In order to answer this question, we have to notice that the technique of music composition and the design of music composition computer programs are in most cases strongly intertwined[4]. Computer software developing/programming and composing music are both complex processes. Both share logic and logical processes as sets of abstract rules which are applied as their' common ground. These sets of rules are either based on processes defined as strict mathematical functions or as musical, mostly (non-)logical historical rules based Harmony, used by consensus: e.g. Jean Phillip Rameau (Rameau 1722).
The inner workings of a computer program and the way it represents itself to the user reflects the thought of the software designer, in our case a composer and to a lesser extent that of the application programmer. Normally the application programmers' role is restricted in 'hammering out' the source code and translate the design concepts and issues into a working application[5].
The ideas and concepts of how to compose music and the presentation (Graphical User Interface = GUI) and behaviour of the software, reflects the original idea or concept of a composer[6] as the principal software designer on the process of creating algorithmic music by using a formal and precise process description which is translated into procedures and functions (algorithms[7]).

By having a certain idea about interacting with the program and taking into account a certain concept about the way how to compose algorithmic/computer music, we could state that indeed we do need (many) more composition programs as they just reflect a limited idea about the complex process of composing music.

BUT WHAT IS CACE4?

As previously stated CACE4 is a data analysing and transforming algorithmic music composition program in one. The abbreviation stands for Computer Aided Composition Environment version 4, and is the follow-up of CAC3, a much earlier and simpler attempt[8] in creating such an application. CACE4 has

[4] From a historical perspective, many algorithmic music composition programs have been developed, just to mention a few: Iannis Xenakis: UPIC (see: http://www.centre-iannis-xenakis.org/cix_upic_presentation?lang=en), Gottfried Michael Koenig: Project 1 & Project 2 (see: http://www.koenigproject.nl/Programmed_Music.pdf), Paul Berg: AC Toolbox (see: http://www.actoolbox.net) and Rick Taube's Grace (see: http://commonmusic.sourceforge.net), are some examples of many more algorithmic compositions programs developed through the years. For a more elaborated list see: Tim Thomson: http://nosuch.com/tjt/plum.html and Paul Doornbush: http://www.doornbusch.net

[5] besides the normal coding the finishing of the program has to be done by nitty gritty finalising tweaking; mostly painstaking and certainly a time-consuming process. In case of the CACE4 program we are talking about the cycle of coding and debugging and is certainly comparable with the error checking of a music score, before printing, publishing and performing can take place.

[6] In CACE4 there is only one software designer, other music composition programs could have been created by more than one person.

[7] An Algorithm is a formal and precise process description mostly used in a computer programming language. (see for more information: https://www.khanacademy.org/computing/computer-science/algorithms/intro-to-algorithms/v/what-are-algorithms).

[8] MacOS 7.x - 8.5, between 1988 - 1993, developed with MCL 4.3.5, created during the authors BA study.

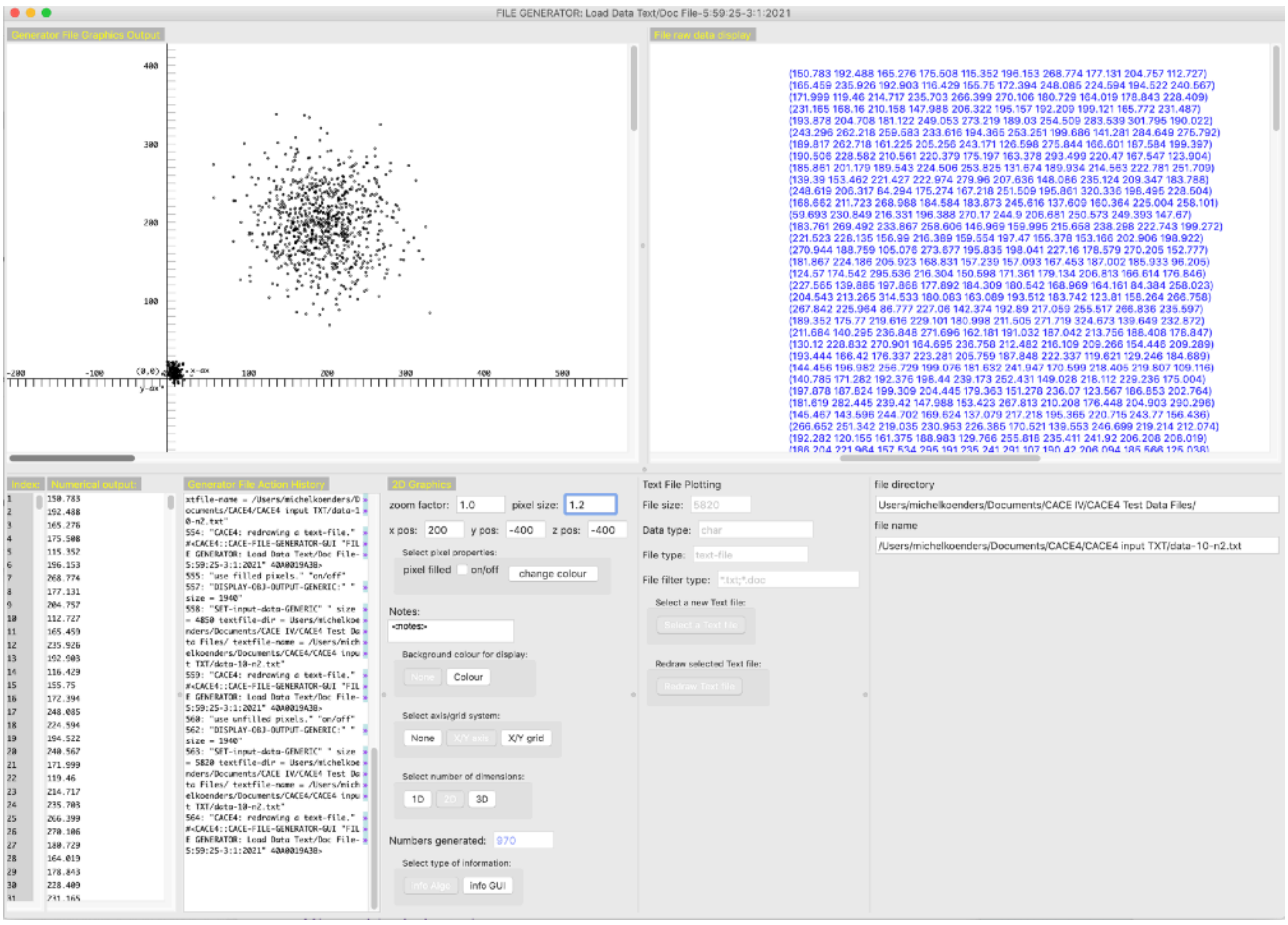
been developed in the computer programming language: List Programming (LISP for short[9]). The way it is, CACE4 is also related to the world of data analysis and statistics as well to the ‘family’ of algorithmic music composition programs. Explicit knowledge about music can only be found in a few separated modules or CACE4 Objects as they are called in CACE4.
Therefore, a closer relation between CACE4 and general data analysis programs can be noticed: the core focus is on analysing data, by either reading a file[10] or making use of several Generator Objects[11] and finding some sort of correlation and structure between the entries. Thus, showing us connectivity and possible (hidden) structures.
And that makes CACE4 a kind of Information Retrieval (IR) toolkit suited for analysing almost any text based data-sets while the other CACE4 Objects are capable of analysing and transforming the data-set for using it in a music composition context.
All these different CACE4 Objects can be linked or chained together in a more or less random order, while the communication between these different Objects is just a single ‘stream’ of numbers[12]. Thus building a strategy to solve a specific approach in data analysis and transforming by using the Translator Objects into music output.

[9] LISP is an all-purpose, Computer Programming language originally created for doing lambda calculus (invented by John McCarthy in 1958). “In fact, the original LISP, introduced by John McCarthy in 1960, known as pure LISP, is completely functional.” (Ghezzi and Jazayeri 1982, 1987, p. 274).

[10] In case of a CACE4 File Generator Object plain text (.txt) files can be read in. Or with the separate CACE4 Spear (Klingbeil 2009) Partials Text File Object: Spear analysis files. Spear analysis files are DSP description files where after analysing an Audio file (with a Short Time Fourier Transforms.) pairs of Amplitude and frequency components are ordered, according their appearance in time (based on the sample rate). For more information see: Spear© created by Michael Klingbeil, is an Analysis/Synthesis program based on the principles of STFT. See http://www.klingbeil.com/spear/ for further details about Spear.

[11] There are several CACE4 Generator Objects. The group is split into Math Generators: with a number of fractal and attractor Generator Objects available for generating data, and File Generators able to read text files (.txt and .doc), SMF (type 0) and SPEAR analysis files. For a full list of all CACE4 Objects see the appendix, pg. 22.

[12] There’s one important exception when an Annotator Object is inserted in the chain of strategy. Then, not only the data is transferred to the next CACE4 Object, but also the added annotation of the data is transferred (see the explanation of the use of a CACE4 Annotation Object: pg. 12).



```
(150.783 192.488 165.276 175.508 115.352 196.153 268.774 177.131 204.757 112.727)
(165.459 235.926 192.903 116.429 155.75 172.394 248.085 224.594 194.522 240.567)
(171.999 119.46 214.717 235.703 266.399 270.106 180.729 164.019 178.843 228.409)
(231.165 168.16 210.158 147.986 206.322 195.157 192.209 199.121 165.772 231.487)
(193.878 204.708 181.122 249.053 273.219 189.03 254.509 283.539 301.795 190.022)
(243.296 262.218 259.583 233.616 194.365 253.251 199.686 141.281 284.649 275.792)
(189.817 262.718 161.225 205.256 243.171 126.598 275.844 166.601 167.584 199.397)
(190.506 228.582 210.561 220.379 175.197 163.378 293.499 220.47 167.547 123.904)
(185.861 201.179 189.543 224.506 253.825 131.674 189.934 214.563 222.781 251.709)
(139.39 153.462 221.427 222.974 279.96 207.636 148.086 235.124 209.347 183.788)
(248.619 206.317 84.294 175.274 167.218 251.509 195.861 320.336 196.495 228.504)
(168.662 211.723 268.988 184.584 183.873 245.616 137.609 160.364 225.004 258.101)
(59.693 230.849 216.331 196.388 270.17 244.9 206.681 250.573 249.393 147.67)
(183.761 269.492 233.867 258.606 146.969 159.995 215.658 238.298 222.743 199.272)
(221.523 228.135 156.99 216.389 159.554 197.47 155.378 153.166 202.906 198.922)
(270.944 188.759 105.078 273.677 195.835 198.041 227.16 178.579 270.205 152.777)
(181.867 224.186 205.923 168.831 157.239 157.093 167.453 187.002 185.933 96.205)
(124.57 174.542 295.536 216.304 150.598 171.361 179.134 206.813 166.614 176.846)
(227.565 139.885 197.866 177.892 184.309 180.542 168.969 164.161 84.384 258.023)
(204.543 213.265 314.533 180.083 163.089 193.512 183.742 123.81 158.264 266.758)
(267.842 225.964 86.777 227.06 142.374 192.89 217.059 255.517 266.836 235.597)
(189.352 175.77 219.616 229.101 180.998 211.505 271.719 324.673 139.649 232.872)
(211.684 140.295 236.848 271.696 162.181 191.032 187.042 213.756 186.408 176.847)
(130.12 228.832 270.901 164.695 236.758 212.482 216.109 209.266 154.446 209.289)
(193.444 166.42 176.337 223.281 205.759 187.848 222.337 119.621 129.246 184.689)
(144.456 196.982 256.729 199.076 181.632 241.947 170.599 218.405 219.807 109.116)
(140.785 171.282 192.376 196.44 239.173 252.431 149.028 216.112 229.236 175.004)
(197.878 187.624 199.309 204.445 179.363 151.278 236.07 123.567 186.853 202.764)
(181.619 282.445 239.42 147.988 153.423 267.813 210.208 176.448 204.903 290.296)
(145.467 143.596 244.702 169.624 137.079 217.218 195.365 220.715 243.77 156.436)
(266.652 251.342 219.035 230.953 226.385 170.521 139.553 246.699 219.214 212.074)
(192.282 120.155 161.375 188.983 129.766 255.818 235.411 241.92 206.208 206.019)
(186.204 221.984 157.534 295.191 235.241 291.107 190.42 206.094 185.566 125.038)
```

Index:	Numerical output:
1	150.783
2	192.488
3	165.276
4	175.508
5	115.352
6	196.153
7	268.774
8	177.131
9	204.757
10	112.727
11	165.459
12	235.926
13	192.903
14	116.429
15	155.75
16	172.394
17	248.085
18	224.594
19	194.522
20	240.567
21	171.999
22	119.46
23	214.717
24	235.703
25	266.399
26	270.106
27	180.729
28	164.019
29	178.843
30	228.409
31	231.165

```
xtfile-name = /Users/michelkoenders/D
ocuments/CACE4/CACE4 input TXT/data-1
0-n2.txt"
554: "CACE4: redrawing a text-file."
#<CACE4::CACE-FILE-GENERATOR-GUI "FIL
E GENERATOR: Load Data Text/Doc File-
5:59:25-3:1:2021" 40A0019A38>
555: "use filled pixels." "on/off"
557: "DISPLAY-OBJ-OUTPUT-GENERIC:" "
size = 1940"
558: "SET-input-data-GENERIC" " size
= 4850 textfile-dir = Users/michelkoe
nders/Documents/CACE IV/CACE4 Test Da
ta Files/ textfile-name = /Users/mich
elkoenders/Documents/CACE4/CACE4 inpu
t TXT/data-10-n2.txt"
559: "CACE4: redrawing a text-file."
#<CACE4::CACE-FILE-GENERATOR-GUI "FIL
E GENERATOR: Load Data Text/Doc File-
5:59:25-3:1:2021" 40A0019A38>
560: "use unfilled pixels." "on/off"
562: "DISPLAY-OBJ-OUTPUT-GENERIC:" "
size = 1940"
563: "SET-input-data-GENERIC" " size
= 5820 textfile-dir = Users/michelkoe
nders/Documents/CACE IV/CACE4 Test Da
ta Files/ textfile-name = /Users/mich
elkoenders/Documents/CACE4/CACE4 inpu
t TXT/data-10-n2.txt"
564: "CACE4: redrawing a text-file."
#<CACE4::CACE-FILE-GENERATOR-GUI "FIL
E GENERATOR: Load Data Text/Doc File-
5:59:25-3:1:2021" 40A0019A38>
```

zoom factor: 1.0 pixel size: 1.2
x pos: 200 y pos: -400 z pos: -400
Select pixel properties:
pixel filled [ ] on/off
Notes: <notes:>
Background colour for display:
Select axis/grid system:
Select number of dimensions:
Numbers generated: 970
Select type of information:

Text File Plotting
File size: 5820
Data type: char
File type: text-file
File filter type: *.txt;*.doc
Select a new Text file:
Redraw selected Text file:

file directory
Users/michelkoenders/Documents/CACE IV/CACE4 Test Data Files/
file name
/Users/michelkoenders/Documents/CACE4/CACE4 input TXT/data-10-n2.txt

Figure 1. A CACE4 File Generator Object displaying data from a plain text file.

CACE4: SOFTWARE DESIGN IDEAS AND APPLIED PROGRAMMING TECHNIQUES.

The initial design criteria of CACE4:
There are four initial goals which acts as major design criteria used for developing the CACE4 application as there are:

1/ As Operating Systems (OS) evolve and change, and the ideas about Music and composition do as well, the CACE4 algorithmic music composition program as such should be an open-ended software application. Fully functional as it is, but not, really finished.
2/ Therefore further development should be done easily with an adaptive predefined modular approach.
3/ CACE4 should be usable for educational purposes as well, with examples and an explanation of the specific procedures followed[13].
4/ And its most important initial design goal: it should be capable for use as an algorithmic composition program.

MVC PARADIGM.

Certain specific software programming models and techniques have been used for developing CACE4 as an application. As there is the Model View Controller (MVC)[14] paradigm software development technique. It is based on a strict division of the programming code into three software components. First of all, a (mathematical) description of the specific technique applied as an algorithm build out of functions and procedure is used for implementing the so-called Model. Secondly the representation of the data is done in a View or window representation. This can either be a text field with a numerical representation or in a graph as 2- or 3-dimensional plotted data.
And thirdly the necessary Controllers are completing this paradigm as there are: menu's, editable text fields, sliders and buttons. These are all software components for letting the user interact with the data and thus actively changing the CACE4 Object behaviour and output.
Transportability of the source code to other, future (different OS) platforms is incorporated into the design of the program. This was achieved by using this strict split between code and GUI, according the MVC paradigm, and doing the LISP coding according ANSI[15] standards as well.

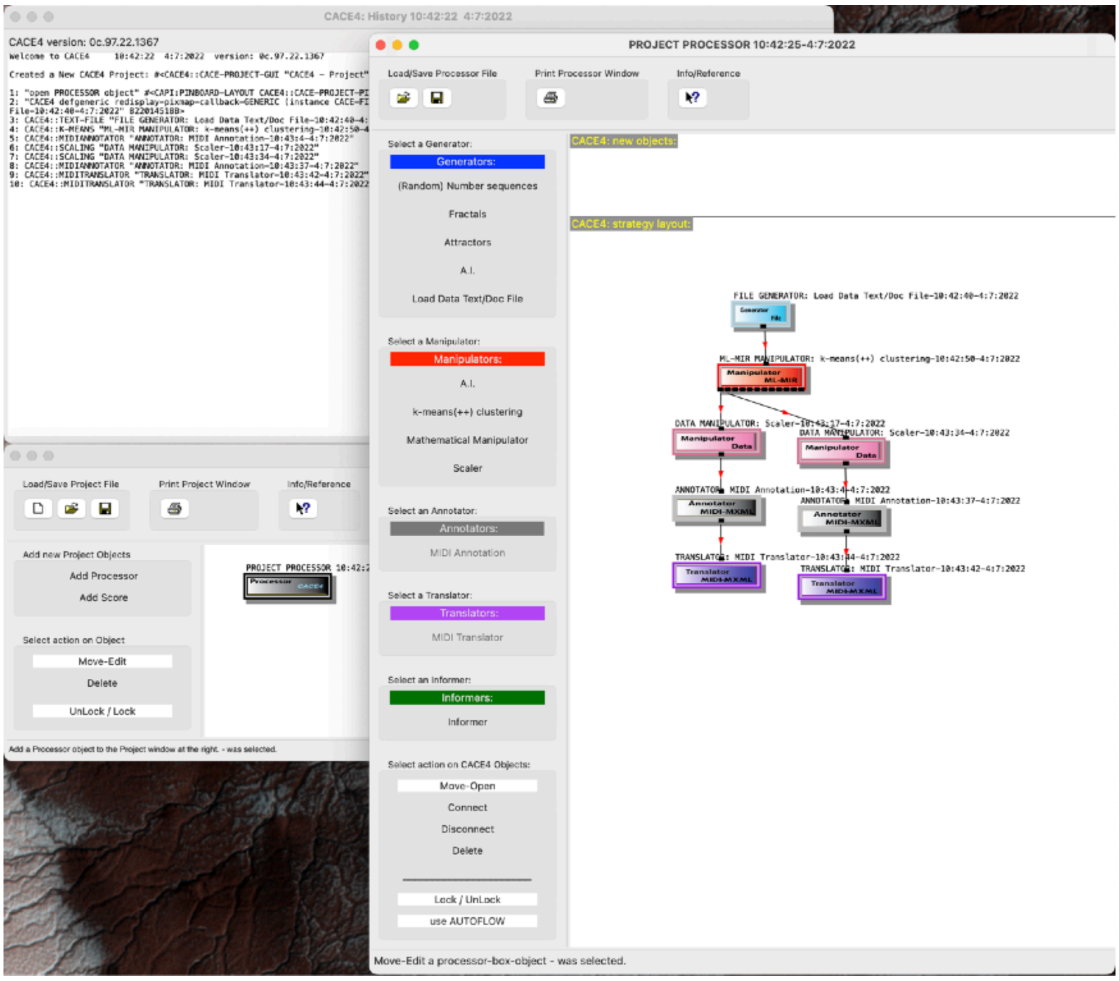
[13] This idea has been incorporated in the GUI of each CACE4 Object as two buttons (title of the button pane: select type of information): info Algo and info GUI. The first one gives a detailed description of the inner workings of the algorithm used. The latter one explains the GUI of the CACE4 Object. Additionally added two more buttons for Wikipedia and Wolfram for extra information about the topics on their specific URL's.

[14] See for more information about MVC: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx

[15] ANSI = American National Standards Institute https://www.ansi.org, The LISP programming is done according ANSI X3J13 1994. And can be found at the ANSi site: https://webstore.ansi.org/Standards/INCITS/ansiincits2261994r2004

COSMOS2.

As previously mentioned CACE4 is a computer program with an easy extendable framework. The use of a framework application design with Object Oriented Programming (OOP) together with a kind of single stream-based IO (= input/output) for communication between two or more CACE4 Objects gives it a flexibility that makes it ready for future adaptations. For the communication between all these separated modules the CACE4 Operating System with the Modular Object Shell # 2 (for short: COSMOS2) has been developed and implemented and is for now sufficient. It is making use of a set of slots shared by all CACE4 Objects for direct communication. It consists of a unique generated Object-ID number[16] and is stored in either the INPUT-LINK slot or OUTPUT-LINK slot of the CACE4 Object. This linking together is visual symbolised by connecting the objects with an arrowed line as a representation of the connection and the direction of the data flow as such (see fig 2, pg. 9).



```
CACE4 version: 0c.97.22.1367
Welcome to CACE4     10:42:22  4:7:2022  version: 0c.97.22.1367

Created a New CACE4 Project: #<CACE4::CACE-PROJECT-GUI "CACE4 - Project"
1: "open PROCESSOR object" #<CAPI:PINBOARD-LAYOUT CACE4::CACE-PROJECT-PI
2: "CACE4 defgeneric redisplay-pixmap-callback-GENERIC (instance CACE-FI
File-10:42:40-4:7:2022" 8220145188>
3: CACE4::TEXT-FILE "FILE GENERATOR: Load Data Text/Doc File-10:42:40-4:
4: CACE4::K-MEANS "ML-MIR MANIPULATOR: k-means(++) clustering-10:42:50-4
5: CACE4::MIDIANNOTATOR "ANNOTATOR: MIDI Annotation-10:43:4-4:7:2022"
6: CACE4::SCALING "DATA MANIPULATOR: Scaler-10:43:17-4:7:2022"
7: CACE4::SCALING "DATA MANIPULATOR: Scaler-10:43:34-4:7:2022"
8: CACE4::MIDIANNOTATOR "ANNOTATOR: MIDI Annotation-10:43:37-4:7:2022"
9: CACE4::MIDITRANSLATOR "TRANSLATOR: MIDI Translator-10:43:42-4:7:2022"
10: CACE4::MIDITRANSLATOR "TRANSLATOR: MIDI Translator-10:43:44-4:7:2022
```

Add a Processor object to the Project window at the right. - was selected.

Move-Edit a processor-box-object - was selected.

Figure 2. A simple CACE4 strategy of chained/linked CACE4 Objects. Notice the four interaction possibilities with the CACE4 Objects: Move-open for moving or editing the selected CACE4 Object. Connect, Disconnect or Delete are the other three interactions of the CACE4 Objects.

[16] This unique Object-ID number is created at the moment an CACE4 Object is added to the process window.

FROM CONCEPT TO A WORKING CACE4 PROGRAM.

As previously stated CACE4 was original designed as an open-ended computer program, what is reflected the way it is constructed according rules and methods belonging to a framework application[17]. This gives way to have a freer and more specific design for every CACE4 Object and to be open ended, based on the principles of a pre-designed framework which doesn't have to be finished in order to be fully functional as an application. More CACE4 Objects can be rather easily added to the existing ones if necessary.

This open framework programming technique is obtained by making use of Object Oriented Programming (OOP) techniques. Making development and adaptation of the application swift and easy. Because CACE4 is programmed in the computer language LISP it makes use of the Common Lisp Object System[18] (CLOS) as an already incorporated LISP Library (in the LispWorks IDE) to achieve this goal. The CLOS is intertwined with the Common Application Programming Interface (CAPI) Library (also in the LispWorks IDE) and therefore it is easy, with an initialise-instance method call and an additional add-views method, to extend basic GUI classes with extra or altered views and controllers, adapted to the specific needs of every CACE4 Object.

ABOUT THE CACE4 GUI.

The CACE4 Object system GUI has an Object-Oriented Programming Style (OOPS) design for both the GUI and the underlying calculations and processes (Model) in a Generator, Manipulator, Annotator, Translator or Informer wrapper class. This wrapper class facilitates two major designs items: one providing numerical input/output display only and the other one: plotting in a 1D, 2D or 3D space.

Although more elaborated and resulting in more coding, this makes it possible to split the functionality of the GUI almost completely from the (underlying) functionality of the software. This idea has been used as a fundamental design approach of the CACE4 program to give every Model which is the underlaying functionality of a CACE4 Object its optimum GUI. Therefore, fine tuning[19] of the GUI can be done for every single CACE4 Object.

[17] A framework application can be defined as an application build out of reusable software components (The application framework). 'Wrapped' in these frames (mostly GUI based), objects can easily be embedded into the application environment as such. See for more details about the topic: http://www1.cse.wustl.edu/~schmidt/CACM-frameworks.html

[18] CLOS is an ANSI Standard, Object Oriented Programming language extension for LISP (1994, ANSI X3J13), as described in Object-Oriented Programming in Common Lisp, A Programmer's Guide to CLOS (Keene 1989), and Understanding CLOS, The Common Lisp Object System (Lawless and Miller 1991).

[19] This is made possible due to the fact that the design is based on the on-board CAPI GUI classes: provided by the LispWorks IDE combining CAPI with CLOS.

CACE4 OBJECT IO.

As previously stated CACE4 is not based on a specific musical language design, a more common design approach for an algorithmic music program[20]. To make the communication between CACE4 Objects in the Processor Window possible: COS&MOS2[21] has been developed. The direct involvement of COS&MOS2 in CACE4 can be noticed in the linking (or chaining) of different CACE4 Objects by connecting their input and output (IO). All CACE4 Objects share this type of IO communication model as a one-dimensional data-set of rational (single-floats/double-floats or integers) numerical values of any type of length. This rather simple design concept also offers the possibility to isolate the processing of the data-set to the CACE4 Object itself. Therefore, local data handling (e.g. preparing the data for further manipulation and/or calculation) can now be achieved according the necessity of the specific process. This concept of separating the input/output from the data manipulation is fully exploited in the development and the design of the different CACE4 objects.

CACE4 Objects can have multiple inputs and multiple outputs. The number of IO's of a CACE4 Object are only based on the specific process it represents. Most processes have only one input, but a few special cases exist with more than one, input-stream (see fig 3, page 11).

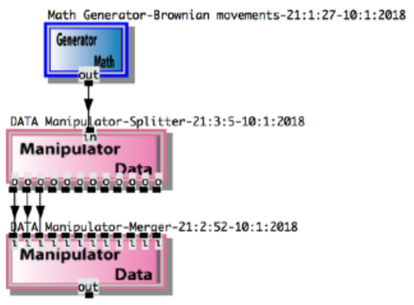

Math Generator-Brownian movements-21:1:27-10:1:2018
Generator
Math
out
DATA Manipulator-Splitter-21:3:5-10:1:2018
in
Manipulator
Data
DATA Manipulator-Merger-21:2:52-10:1:2018
Manipulator
Data
out

Figure 3. Three CACE4 objects with single and multiple input and output connections.

After processing the (input) data-set, the now altered data-set is available at the CACE4 Objects output. In a few special cases several different outputs are available (see fig. 3, pg. 11).

By taking all these different design concepts into account and by making a design based on frames represented by boxes with input and output connections each box acts as a separate process with its specific GUI: optimised and well suited for the process it represents. To make future programming and further development easy; every CACE4 Object GUI is based on either one of two distinct types of GUI designs embedded in CLOS/CAPI classes. One with a graph pane where IO also can be plotted in several ways as 1D, 2D or 3D pixel or function plots and as numerical lists of input and output. The other GUI type is much simpler: it basically has only numerical lists for IO.

[20] E.g. AthenaCL by Christopher Arazi (2005) and Grace by Rick Taube (see footnote 3).

[21] COS&MOS2 stands for CACE Operating System & Modelling and Organising Shell #2

CACE4 based on these separate, boxed processes should be useful not only as data analysers and processors in a musical composition environment but act as educational software as well[22]. The CACE4 Objects offers the opportunity to explain certain processes. Initially separated from other areas and to explain the inner workings to students as isolated and unique processes. E.g. *k*-means/*k*-means++ can be presented as a cluster detection process without the knowledge how to construct a SMF (= Standard MIDI Files). The latter technique of how to construct these SMFs can be if necessary explained to the students as a separate topic at a later date.

For using CACE4 as an algorithmic composition program the 55 (and still growing) different CACE4 Objects offer a large variety of interesting analysing and processing tools. To name just a few: several statistical processes as mean, median, standard deviation, correlation calculation (Pearson, Spearman and and 3 types of Kendall-tau), histograms and linear regression. Besides these more common statistical tools more elaborated processes are available as CACE4 Objects as well. For now (version 0b.97.x) Hierarchical Cluster Techniques and Hierarchical Cluster Analysis (HCT/HCA) as part of Machine Learning techniques (used Linkage methods: WPGMA[23],UPGMA, UPGMC, SL and CL) together with *k*-means, *k*-means++, *k*-Nearest Neighbour (*k*-NN), Expectation Maximisation (EM) and a neural network as 'bottom up' AI process[24]; the Adaptive Resonance Theory #2 (ART2)[25] (Carpenter and Grossberg 1987), and finally a Fuzzy Logic Controller are implemented.

Many other processes as calculation, sorting, filtering, pruning (deleting), merging and scaling complement the data analysing and processing power of CACE4. For an extended overview of all these objects and their specific possibilities take a look at the appendix A (pg. 23). These previous Objects can also be used as tools for data mining: cleaning up the data-set or transforming the input data-set into certain dimensions necessary for doing our calculus.[26]

[22] The author was (up-to June 2021) a free-lance lecturer of students in Computer Music Software Development, mainly in the area of creating Music software applications with a focus on using AI and MIR techniques (LISP) and to a lesser extend in Digital Signal Processing (DSP, C/C++). The students were undergraduates in their second year or fourth year of education and Master students at the University of the Arts, Utrecht the Netherlands.

[23] WPGMA = Weighted Pair Group Method with Arithmetic Mean, UPGMA = Unweighted Pair Group Method with Arithmetic Mean, UPGMC = Unweighted Pair Group Method with Arithmetic Centroid, SL = Simple Linkage and CL = Complete Linkage. All bottom-up Hierarchical Cluster Analysis Linkage methods. All these methods are all implemented in their naive format and don't make use of optimised algorithms. Therefore calculation of larger data-sets can be time consuming.

[24] bottom up AI approach, also connectionist model takes a model of our neural patterns as they have developed in our brains. These networks have no a priori knowledge about, in this case music, but act as a true, although abstract, mirror of the workings of how brain cells are connected and how they cooperate together in transferring messages in these networks. See for more information: https://en.wikipedia.org/wiki/Connectionism

[25] ART2 Neural Network, is a Neural Network based on the use of Adaptive Resonance Theory originally designed and proposed by Carpenter & Grossberg (Carpenter and Grossberg 1987). See for more information: https://nl.mathworks.com/matlabcentral/fileexchange/54377-art-2-neural-network---machine-monitoring-hybrid-system?s_tid=gn_loc_drop or https://en.wikipedia.org/wiki/Adaptive_resonance_theory or https://web.archive.org/web/20120109162743/http://users.visualserver.org/xhudik/art

[26] In future version of CACE4 a separate data mining Object will be added to deal with the specific features of Data Mining. (e.g. Analysing and pre-processing of the acquired data-set.

There are two slightly different CACE4 Objects; the Annotator and the Translator Objects. With these two CACE4 Objects it is also possible to first annotate and later use the annotated input to translate it into a more musical pre-MIDI format. In the present version, not all CACE4 Objects are capable of processing this annotated data (future versions will add more CACE4 Objects to this list), but 9 can (see the appendix A, pg. 23). The CACE4 Translator Object must be used as the last CACE4 Object in the chain as part of a designed strategy (this will be further explained in the next paragraph). Its output can be saved after processing in a plain Text file or send to the CACE4 Score Object where the translated data can be written into a file by creating a SMF[27], type 0.

DEVELOPING A STRATEGY BY CHAINING CACE4 OBJECTS INTO A PROCESS.

In order to be able to mix IR techniques with other, statistical aspects of data (as such) and turn them into Music related data (e.g. for now MIDI and later MXML). It is necessary to use common ground: in this case Mathematics is preferable above an (abstract) musical language: which could introduce 'false' connections, by introducing other hierarchies, as defined in a specific language before any IR techniques are used.

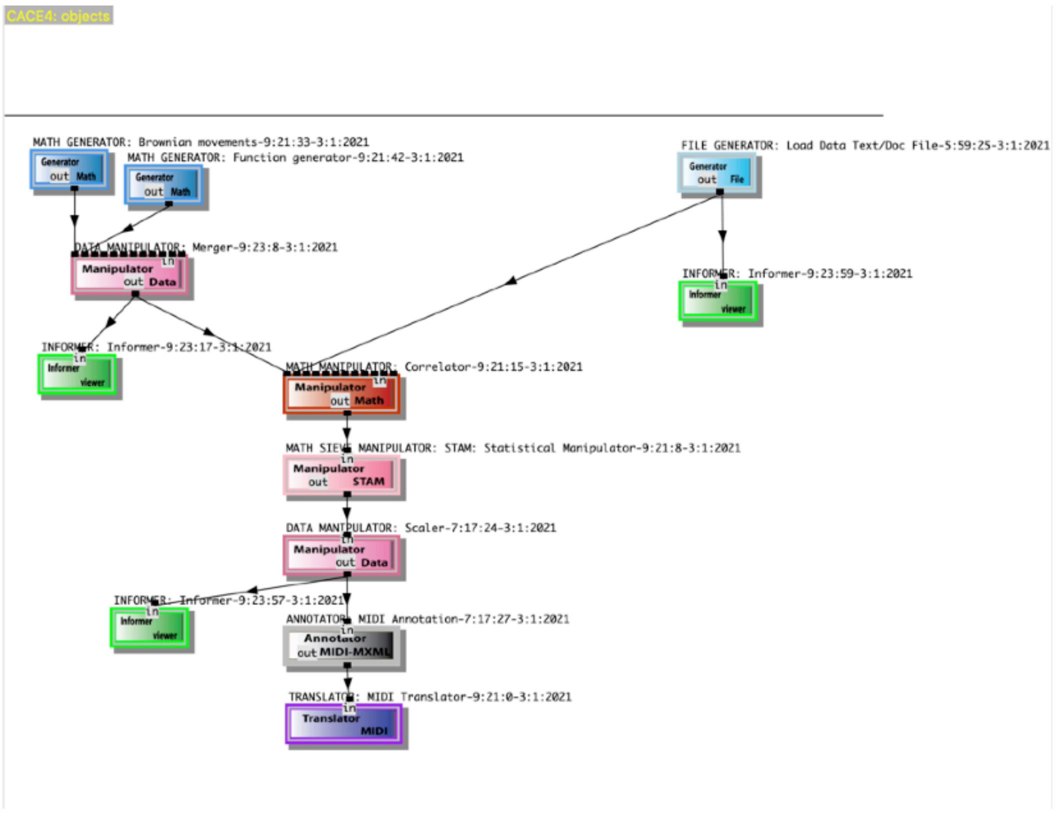

CACE4: objects
MATH GENERATOR: Brownian movements-9:21:33-3:1:2021
Generator out Math
MATH GENERATOR: Function generator-9:21:42-3:1:2021
Generator out Math
FILE GENERATOR: Load Data Text/Doc File-5:59:25-3:1:2021
Generator out File
DATA MANIPULATOR: Merger-9:23:8-3:1:2021
Manipulator in out Data
INFORMER: Informer-9:23:59-3:1:2021
in Informer viewer
INFORMER: Informer-9:23:17-3:1:2021
in Informer viewer
MATH MANIPULATOR: Correlator-9:21:15-3:1:2021
Manipulator in out Math
MATH SIEVE MANIPULATOR: STAM: Statistical Manipulator-9:21:8-3:1:2021
in Manipulator out STAM
DATA MANIPULATOR: Scaler-7:17:24-3:1:2021
in Manipulator out Data
INFORMER: Informer-9:23:57-3:1:2021
in Informer viewer
ANNOTATOR: MIDI Annotation-7:17:27-3:1:2021
in Annotator out MIDI-MXML
TRANSLATOR: MIDI Translator-9:21:0-3:1:2021
in Translator MIDI

Figure 4. Example of a CACE4 Processor Object which displays a more elaborated strategy. A (Brownian) Random is combined with a function, merged together and correlated with a data file. Then we use several statistical techniques to obtain a new dataset and scale the values in the MIDI domain. After scaling we annotate it and translate it into the MIDI data which can be 'send' to the score object in a separate Project window: together with the Processor Object where it can be written into a SMF (type 0).

[27] SMF type 2 and Music-XML (MXML) format will be available soon.

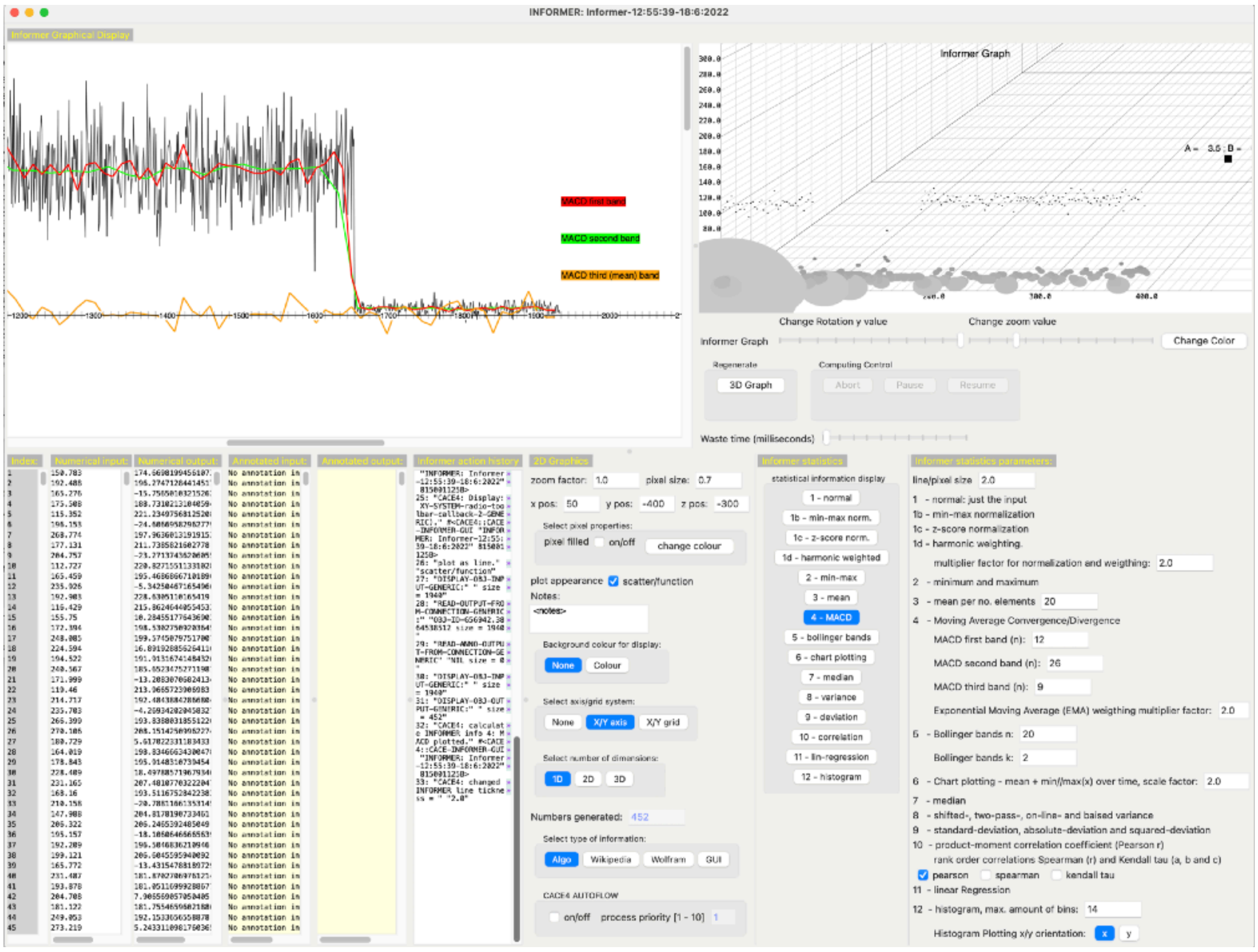
This concept doesn't free us of the introduction of any music language but the concept as such is transferred to an isolated CACE4 Object (hence our tandem of CACE4 Annotator & Translator Objects are such Objects). Data handling is now strictly done according a specific design of the object and its use in a strategy in a CACE4 Processor Object (see figure 2 and figure 4 for a more elaborated CACE4 strategy example).

Every CACE4 Object, after modifying or transforming it's input according its process description is presenting its output as a single numerical, one dimensional data-set, ready for attachment to other CACE4 objects in the CACE4 Processor window as part of a desired strategy. The only exceptions to this approach are the Annotator and Translator Objects, which work in tandem to add extra, by the user annotated information about the 1-dimensional data-set between the CACE4 Objects. They have no further influence on the data or further processing as such but are only used to add extra information about the number in a so-called LISP hash-table construction[28]. All these different boxes can be chained or linked together to form more complex processes in order to analyse and transform the original input data. Consequently, unique processes can be created by the user. These strategies as chained CACE4 Objects can be altered and saved to file, and read in at a later stage, for further use.

As an extra data analysing and visualisation Object, the Informer Object[29] in green (see figure 5, page 15) can be attached to any output without altering the data-set of that CACE Object. With the Informer Object comes a whole range of statistics: e.g. minimum-maximum, mean, median, Moving Average Convergence/Divergence (MACD), Bollinger Bands, chart plotting, variance, deviation, correlation (three different types: Pearson (product moment), Spearman (rank-order) and Kendall tau (a, b and c), Linear Regression and Histograms can be used.

[28] A LISP hash-table is a special data storage type (just like floats, integers or structures) optimised for storing large amounts of data is a specific format. This LISP hash-table is constructed as a 4-member table, consisting of the number itself either as an integer, rational, complex, single-float or as a double-float. The other 3 places containing text as a single string with values on micro-, meso- and macro-level as added by the user. These values will be later on used by the CACE4 Translator object, as the final object in our sequence of CACE4 objects (strategy chain), to translate the 1-dimensional stream of numerical values into corresponding MIDI/MXML values.

[29] The Informer Objects are the green coloured objects in figure 4, pg. 13.


INFORMER: Informer-12:55:39-18:6:2022

2D Graphics

zoom factor: 1.0 pixel size: 0.7

x pos: 50 y pos: -400 z pos: -300

Select pixel properties: pixel filled [ ] on/off

plot appearance [✓] scatter/function

Notes: <notes>

Background colour for display: [✓ None] Colour

Select axis/grid system: None [✓ X/Y axis] X/Y grid

Select number of dimensions: [✓ 1D] 2D 3D

Numbers generated: 452

Select type of information: [✓ Algo] Wikipedia Wolfram GUI

CACE4 AUTOFLOW [ ] on/off process priority [1 - 10] 1

Informer statistics

statistical information display: 1 - normal, 1b - min-max norm., 1c - z-score norm., 1d - harmonic weighted, 2 - min-max, 3 - mean, [✓ 4 - MACD], 5 - bollinger bands, 6 - chart plotting, 7 - median, 8 - variance, 9 - deviation, 10 - correlation, 11 - lin-regression, 12 - histogram

Informer statistics parameters:

line/pixel size 2.0

1 - normal: just the input
1b - min-max normalization
1c - z-score normalization
1d - harmonic weighting.
multiplier factor for normalization and weigthing: 2.0
2 - minimum and maximum
3 - mean per no. elements 20
4 - Moving Average Convergence/Divergence
MACD first band (n): 12
MACD second band (n): 26
MACD third band (n): 9
Exponential Moving Average (EMA) weigthing multiplier factor: 2.0
5 - Bollinger bands n: 20
Bollinger bands k: 2
6 - Chart plotting - mean + min//max(x) over time, scale factor: 2.0
7 - median
8 - shifted-, two-pass-, on-line- and baised variance
9 - standard-deviation, absolute-deviation and squared-deviation
10 - product-moment correlation coefficient (Pearson r)
rank order correlations Spearman (r) and Kendall tau (a, b and c)
[✓] pearson [ ] spearman [ ] kendall tau
11 - linear Regression
12 - histogram, max. amount of bins: 14
Histogram Plotting x/y orientation: [✓ x] y

Figure 5. Example of a CACE4 Informer Object displaying, in colour the Moving Average Convergence/Divergence (MACD) of the input data-set plotted in black (Numerical Input). The calculated result (Numerical Output) can be displayed in several ways. As a function or as a scatterplot of the input data-set (lefthand site). And the calculated output of the MACD can be displayed as a scatterplot in a quasi 3D setting (righthand sight) as well.

Therefore, each strategy for analysing data by using specific analysis techniques or for creating an algorithmic composition, can be done in the context of the available CACE4 Objects with all their unique features. The CACE4 Object order (in the chain of CACE4 Objects) can be altered: new ones can be added and already existing one's deleted or moved and reconnected when necessary.

The only other conceptual restrictions are that every strategy chain needs at least one Generator Object in the beginning. This is a special CACE4 Object with only one output[30] (see fig. 4, pg. 13 and fig. 6, pg. 16) and it also needs a CACE4 Translator Object (with only input and no direct output[31]) at the end of this

[30] The CACE4 Generator Objects come in two flavours: as a CACE4 MATH Generator, which represents a fractal, attractor or other function calculation. Or as a CACE4 File Generator which is capable of reading one of these three formats: a plain text file, a SMF type 0 or a SPEAR analysis text file.

[31] These Transformer Objects as they are called do have a 'hidden' output. After transforming the data in a pre-MIDI context (start-time pitch velocity and duration) it is possible to send it to the Score OBJ. This specific OBJ does the final translation of all selected data, with all the necessary MIDI calculations needed (e.g. MIDI time stamping, tempo and if selected transforming from absolute time to delta time), to turn it into a real SMF type 0.

strategy chain. In between all other Manipulators Objects can be used in any order: they have both one or more inputs and one or more outputs.

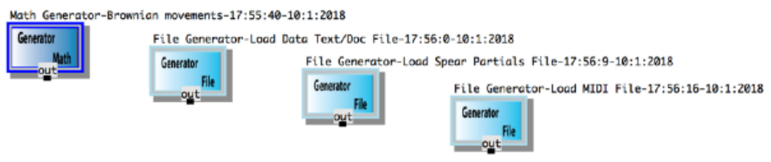

Math Generator-Brownian movements-17:55:40-10:1:2018
Generator
Math
out
File Generator-Load Data Text/Doc File-17:56:0-10:1:2018
Generator
File
out
File Generator-Load Spear Partials File-17:56:9-10:1:2018
Generator
File
out
File Generator-Load MIDI File-17:56:16-10:1:2018
Generator
File
out

Figure 6. Some of the CACE4 Generator Objects. In total there are 23 different Generator Objects. See Appendix 1 for a complete list of all available CACE4 Objects: version 0.97.x

CACE4: THE USE AS A COMPUTER MUSIC COMPOSITION PROGRAM.

CACE4 can be used to analyse data-sets only but this is not its major goal[32] and is therefore not recommended. The name of CACE4[33] points directly to the use of assistance offered in creating algorithmic music compositions. These generated or algorithmic computer compositions can either be pure electronic works: e.g. by generating MIDI Controller data or MIDI notes for use in any Digital Audio Editor and MIDI sequencer program[34], or for use as a starting point for instrumental scores for Music scoring programs as Sibelius[35] and Finale[36] which can be played at a later time by Musicians. The basic method of generating musical material is the same. By creating a strategy as previous described we can design our own algorithmic composition process. In case of an electronic (MIDI) composition generating data for a MIDI controller and combined with MIDI note values can be sufficient and can be done by using the CACE4 Annotator Object and selecting the right MIDI command for annotating the data on Micro level (see fig 7, pg. 17).

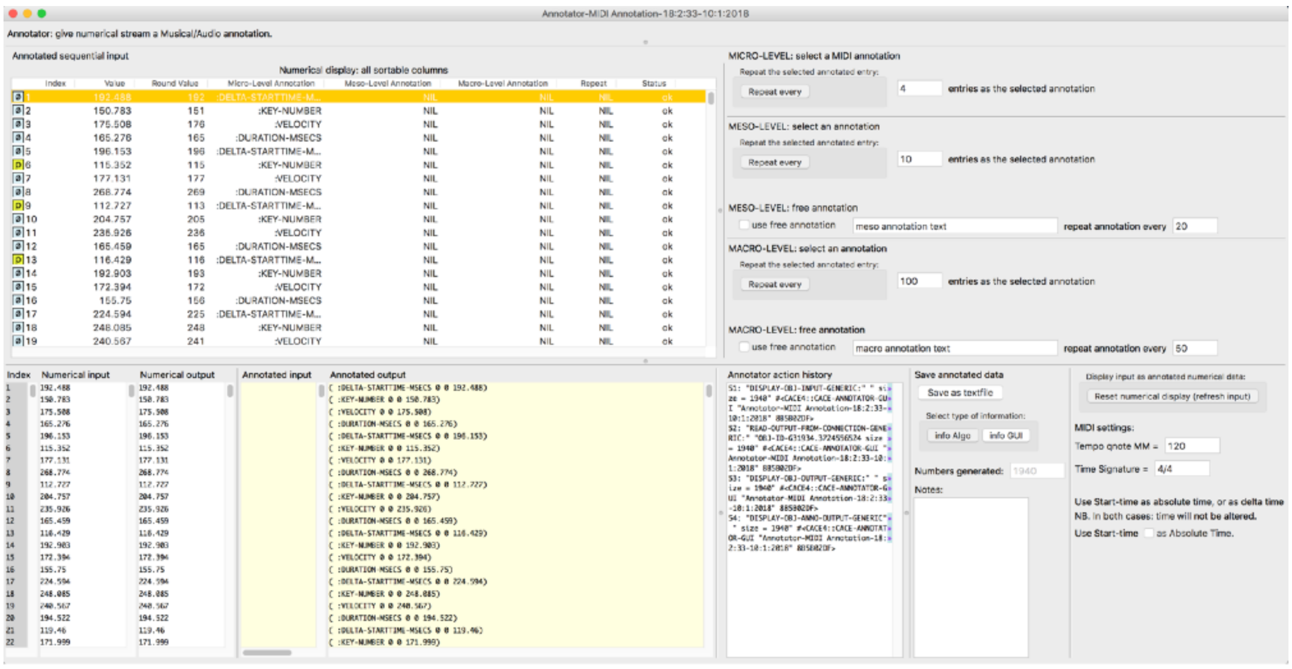
[32] Except in some cases where only analyses of certain data-sets is done for MIR . In mathematical data analysis only it is recommended to use the free R-Project: R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. URL: https://www.r-project.org

[33] CACE4 = Computer Aided Composition Environment #4.

[34] These files are generated in the CACE4 Score Object as SMF (format 0), and can be read by any Sequencer Editor program: Digital Performer (see: http://www.motu.com) or GarageBand from Apple or Logic Pro X or Music Notation Software e.g. Finale from makemusic: https://www.finalemusic.com

[35] See: http://www.avid.com/sibelius for more information.

[36] See: https://www.finalemusic.com/products/ for more information.


Annotator-MIDI Annotation-18:2:33-10:1:2018

Annotator: give numerical stream a Musical/Audio annotation.

Annotated sequential input

Numerical display: all sortable columns

Index	Value	Round Value	Micro-Level Annotation	Meso-Level Annotation	Macro-Level Annotation	Repeat	Status
1	192.488	192	:DELTA-STARTTIME-M...	NIL	NIL	NIL	ok
2	150.783	151	:KEY-NUMBER	NIL	NIL	NIL	ok
3	175.508	176	:VELOCITY	NIL	NIL	NIL	ok
4	165.276	165	:DURATION-MSECS	NIL	NIL	NIL	ok
5	196.153	196	:DELTA-STARTTIME-M...	NIL	NIL	NIL	ok
6	115.352	115	:KEY-NUMBER	NIL	NIL	NIL	ok
7	177.131	177	:VELOCITY	NIL	NIL	NIL	ok
8	268.774	269	:DURATION-MSECS	NIL	NIL	NIL	ok
9	112.727	113	:DELTA-STARTTIME-M...	NIL	NIL	NIL	ok
10	204.757	205	:KEY-NUMBER	NIL	NIL	NIL	ok
11	235.926	236	:VELOCITY	NIL	NIL	NIL	ok
12	165.459	165	:DURATION-MSECS	NIL	NIL	NIL	ok
13	116.429	116	:DELTA-STARTTIME-M...	NIL	NIL	NIL	ok
14	192.903	193	:KEY-NUMBER	NIL	NIL	NIL	ok
15	172.394	172	:VELOCITY	NIL	NIL	NIL	ok
16	155.75	156	:DURATION-MSECS	NIL	NIL	NIL	ok
17	224.594	225	:DELTA-STARTTIME-M...	NIL	NIL	NIL	ok
18	248.085	248	:KEY-NUMBER	NIL	NIL	NIL	ok
19	240.567	241	:VELOCITY	NIL	NIL	NIL	ok

MICRO-LEVEL: select a MIDI annotation
Repeat the selected annotated entry: Repeat every 4 entries as the selected annotation

MESO-LEVEL: select an annotation
Repeat the selected annotated entry: Repeat every 10 entries as the selected annotation

MESO-LEVEL: free annotation
use free annotation: meso annotation text — repeat annotation every 20

MACRO-LEVEL: select an annotation
Repeat the selected annotated entry: Repeat every 100 entries as the selected annotation

MACRO-LEVEL: free annotation
use free annotation: macro annotation text — repeat annotation every 50

Index	Numerical input	Numerical output	Annotated input	Annotated output
1	192.488	192.488		(:DELTA-STARTTIME-MSECS 0 0 192.488)
2	150.783	150.783		(:KEY-NUMBER 0 0 150.783)
3	175.508	175.508		(:VELOCITY 0 0 175.508)
4	165.276	165.276		(:DURATION-MSECS 0 0 165.276)
5	196.153	196.153		(:DELTA-STARTTIME-MSECS 0 0 196.153)
6	115.352	115.352		(:KEY-NUMBER 0 0 115.352)
7	177.131	177.131		(:VELOCITY 0 0 177.131)
8	268.774	268.774		(:DURATION-MSECS 0 0 268.774)
9	112.727	112.727		(:DELTA-STARTTIME-MSECS 0 0 112.727)
10	204.757	204.757		(:KEY-NUMBER 0 0 204.757)
11	235.926	235.926		(:VELOCITY 0 0 235.926)
12	165.459	165.459		(:DURATION-MSECS 0 0 165.459)
13	116.429	116.429		(:DELTA-STARTTIME-MSECS 0 0 116.429)
14	192.903	192.903		(:KEY-NUMBER 0 0 192.903)
15	172.394	172.394		(:VELOCITY 0 0 172.394)
16	155.75	155.75		(:DURATION-MSECS 0 0 155.75)
17	224.594	224.594		(:DELTA-STARTTIME-MSECS 0 0 224.594)
18	248.085	248.085		(:KEY-NUMBER 0 0 248.085)
19	240.567	240.567		(:VELOCITY 0 0 240.567)
20	194.522	194.522		(:DURATION-MSECS 0 0 194.522)
21	119.46	119.46		(:DELTA-STARTTIME-MSECS 0 0 119.46)
22	171.999	171.999		(:KEY-NUMBER 0 0 171.999)

Annotator action history

```
S1: "DISPLAY-OBJ-INPUT-GENERIC:" " si»
ze = 1940" #<CACE4::CACE-ANNOTATOR-GU»
I "Annotator-MIDI Annotation-18:2:33-»
10:1:2018" 8858020F>
S2: "READ-OUTPUT-FROM-CONNECTION-GENE»
RIC:" "OBJ-ID-G31934.3724556524 size »
= 1940" #<CACE4::CACE-ANNOTATOR-GUI "»
Annotator-MIDI Annotation-18:2:33-10:»
1:2018" 8858020F>
S3: "DISPLAY-OBJ-OUTPUT-GENERIC:" " s»
ize = 1940" #<CACE4::CACE-ANNOTATOR-G»
UI "Annotator-MIDI Annotation-18:2:33»
-10:1:2018" 8858020F>
S4: "DISPLAY-OBJ-ANNO-OUTPUT-GENERIC"»
 " size = 1940" #<CACE4::CACE-ANNOTAT»
OR-GUI "Annotator-MIDI Annotation-18:»
2:33-10:1:2018" 8858020F>
```

Save annotated data
Save as textfile
Select type of information: info Algo | info GUI
Numbers generated: 1940
Notes:

Display input as annotated numerical data:
Reset numerical display (refresh input)

MIDI settings:
Tempo qnote MM = 120
Time Signature = 4/4

Use Start-time as absolute time, or as delta time
NB. In both cases: time will not be altered.
Use Start-time ☐ as Absolute Time.

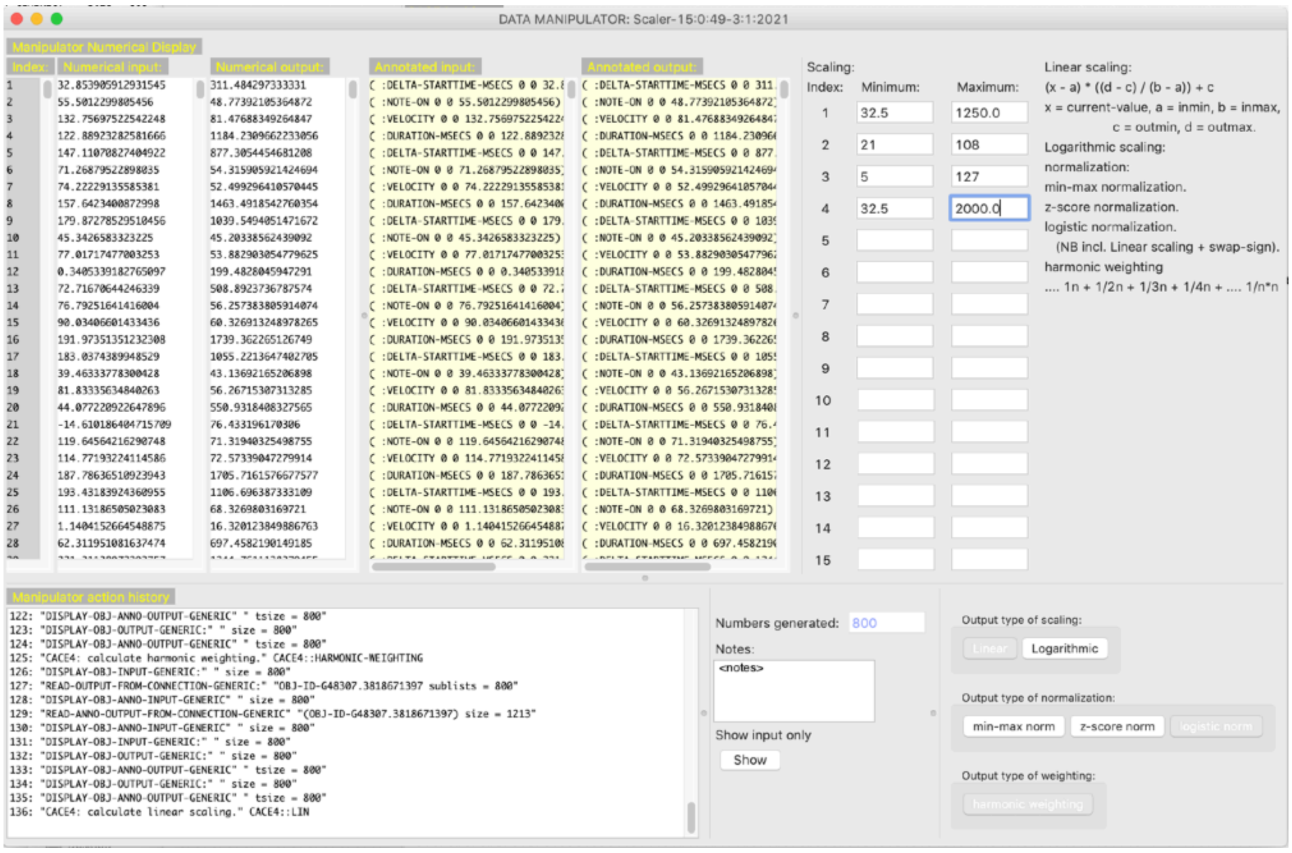
Figure 7. The CACE4 Annotator Object with annotated data values for the micro-level.

To generate Scores we have to use the CACE4 Scaler Object (see fig. 8, pg. 18) in order to map (in this case) linear data values inside the MIDI domain boundaries' desired. For most data: between [0/1,127] as there are pitch and velocity. For use in the domain of time we have to stretch the upper limits. Be aware that milliseconds timing is done in so called delta-timing or inter onset timing in case of the start time (position) for the note and/or MIDI controller. This timing in milliseconds is also used for calculating the duration of the note. See table 1 (pg. 17) for values of this timing used by tempo MM 120 and time signature 4/4[37].

Note durations	in 1/1000 sec	rounded msecs.
1/128	15.625	15.6
1/128 dot	23.4375	23.4
1/64	31.25	31.3
1/64 dot	46.875	46.9
1/32	62.5	62.5
1/32 dot	93.75	93.8
1/16	125	125
1/16 dot	187.5	187.5
1/8	250	250
1/8 dot	375	375
1/4	500	500
1/4 dot	750	750
1/2	1000	1000
1/2 dot	1500	1500
1 (whole)	2000	2000
1 dot	3000	3000
2	4000	4000
3	6000	6000
4	8000	8000
5	10000	10000

Table 1. Values of milliseconds timing used by tempo MM 120 and time signature 4/4.

[37] This is a linear process but other tempi can be easily calculated: e.g. for tempo 60 all note duration values need to be multiplied by 2 to obtain the correct durations. Tables are provided in the CACE4 program: menu item: CACE4 References select: CACE4 Note durations Range table.



DATA MANIPULATOR: Scaler-15:0:49-3:1:2021

Manipulator Numerical Display

Index:	Numerical input:	Numerical output:	Annotated input:	Annotated output:
1	32.853905912931545	311.484297333331	(:DELTA-STARTTIME-MSECS 0 0 32.8	(:DELTA-STARTTIME-MSECS 0 0 311.
2	55.5012299805456	48.77392105364872	(:NOTE-ON 0 0 55.5012299805456)	(:NOTE-ON 0 0 48.77392105364872)
3	132.75697522542248	81.47688349264847	(:VELOCITY 0 0 132.756975225422	(:VELOCITY 0 0 81.4768834926484
4	122.88923282581666	1184.2309662233056	(:DURATION-MSECS 0 0 122.889232	(:DURATION-MSECS 0 0 1184.23096
5	147.11070827404922	877.3054454681208	(:DELTA-STARTTIME-MSECS 0 0 147.	(:DELTA-STARTTIME-MSECS 0 0 877.
6	71.26879522898035	54.315905921424694	(:NOTE-ON 0 0 71.26879522898035)	(:NOTE-ON 0 0 54.3159059214246
7	74.22229135585381	52.499296410570445	(:VELOCITY 0 0 74.2222913558538	(:VELOCITY 0 0 52.4992964105704
8	157.6423400872998	1463.4918542760354	(:DURATION-MSECS 0 0 157.64234	(:DURATION-MSECS 0 0 1463.49185
9	179.87278529510456	1039.5494051471672	(:DELTA-STARTTIME-MSECS 0 0 179.	(:DELTA-STARTTIME-MSECS 0 0 103
10	45.3426583323225	45.20338562439092	(:NOTE-ON 0 0 45.3426583323225)	(:NOTE-ON 0 0 45.20338562439092)
11	77.01717477003253	53.882903054779625	(:VELOCITY 0 0 77.0171747700325	(:VELOCITY 0 0 53.8829030547796
12	0.3405339182765097	199.4828045947291	(:DURATION-MSECS 0 0 0.34053391	(:DURATION-MSECS 0 0 199.48280
13	72.71670644246339	508.8923736787574	(:DELTA-STARTTIME-MSECS 0 0 72.7	(:DELTA-STARTTIME-MSECS 0 0 508.
14	76.79251641416004	56.257383805914074	(:NOTE-ON 0 0 76.79251641416004)	(:NOTE-ON 0 0 56.2573838059140
15	90.03406601433436	60.326913248978265	(:VELOCITY 0 0 90.0340660143343	(:VELOCITY 0 0 60.3269132489782
16	191.97351351232308	1739.362265126749	(:DURATION-MSECS 0 0 191.973513	(:DURATION-MSECS 0 0 1739.36226
17	183.0374389948529	1055.2213647402705	(:DELTA-STARTTIME-MSECS 0 0 183.	(:DELTA-STARTTIME-MSECS 0 0 105
18	39.46333778300428	43.13692165206898	(:NOTE-ON 0 0 39.46333778300428)	(:NOTE-ON 0 0 43.13692165206898)
19	81.83335634840263	56.26715307313285	(:VELOCITY 0 0 81.8333563484026	(:VELOCITY 0 0 56.2671530731328
20	44.07722092264789 6	550.9318408327565	(:DURATION-MSECS 0 0 44.077220	(:DURATION-MSECS 0 0 550.93184
21	-14.610186404715709	76.433196170306	(:DELTA-STARTTIME-MSECS 0 0 -14.	(:DELTA-STARTTIME-MSECS 0 0 76.4
22	119.64564216290748	71.31940325498755	(:NOTE-ON 0 0 119.645642162907	(:NOTE-ON 0 0 71.31940325498755)
23	114.77193224114586	72.57339047279914	(:VELOCITY 0 0 114.771932241145	(:VELOCITY 0 0 72.5733904727991
24	187.78636510923943	1705.7161576677577	(:DURATION-MSECS 0 0 187.78636	(:DURATION-MSECS 0 0 1705.71615
25	193.43183924360955	1106.696387333109	(:DELTA-STARTTIME-MSECS 0 0 193.	(:DELTA-STARTTIME-MSECS 0 0 110
26	111.13186505023083	68.3269803169721	(:NOTE-ON 0 0 111.131865050230	(:NOTE-ON 0 0 68.3269803169721)
27	1.140415266454887 5	16.320123849886763	(:VELOCITY 0 0 1.14041526645488	(:VELOCITY 0 0 16.3201238498867
28	62.311951081637474	697.4582190149185	(:DURATION-MSECS 0 0 62.311951	(:DURATION-MSECS 0 0 697.45821

Scaling:

Index:	Minimum:	Maximum:
1	32.5	1250.0
2	21	108
3	5	127
4	32.5	2000.0
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Linear scaling:
(x - a) * ((d - c) / (b - a)) + c
x = current-value, a = inmin, b = inmax,
c = outmin, d = outmax.
Logarithmic scaling:
normalization:
min-max normalization.
z-score normalization.
logistic normalization.
(NB incl. Linear scaling + swap-sign).
harmonic weighting
.... 1n + 1/2n + 1/3n + 1/4n + ... 1/n*n

Manipulator action history

```
122: "DISPLAY-OBJ-ANNO-OUTPUT-GENERIC" " tsize = 800"
123: "DISPLAY-OBJ-OUTPUT-GENERIC:" " size = 800"
124: "DISPLAY-OBJ-ANNO-OUTPUT-GENERIC" " tsize = 800"
125: "CACE4: calculate harmonic weighting." CACE4::HARMONIC-WEIGHTING
126: "DISPLAY-OBJ-INPUT-GENERIC:" " size = 800"
127: "READ-OUTPUT-FROM-CONNECTION-GENERIC" "OBJ-ID-G48307.3818671397 sublists = 800"
128: "DISPLAY-OBJ-ANNO-INPUT-GENERIC" " size = 800"
129: "READ-ANNO-OUTPUT-FROM-CONNECTION-GENERIC" "(OBJ-ID-G48307.3818671397) size = 1213"
130: "DISPLAY-OBJ-ANNO-INPUT-GENERIC" " size = 800"
131: "DISPLAY-OBJ-INPUT-GENERIC:" " size = 800"
132: "DISPLAY-OBJ-OUTPUT-GENERIC:" " size = 800"
133: "DISPLAY-OBJ-ANNO-OUTPUT-GENERIC" " tsize = 800"
134: "DISPLAY-OBJ-OUTPUT-GENERIC:" " size = 800"
135: "DISPLAY-OBJ-ANNO-OUTPUT-GENERIC" " tsize = 800"
136: "CACE4: calculate linear scaling." CACE4::LIN
```

Numbers generated: 800

Notes: <notes>

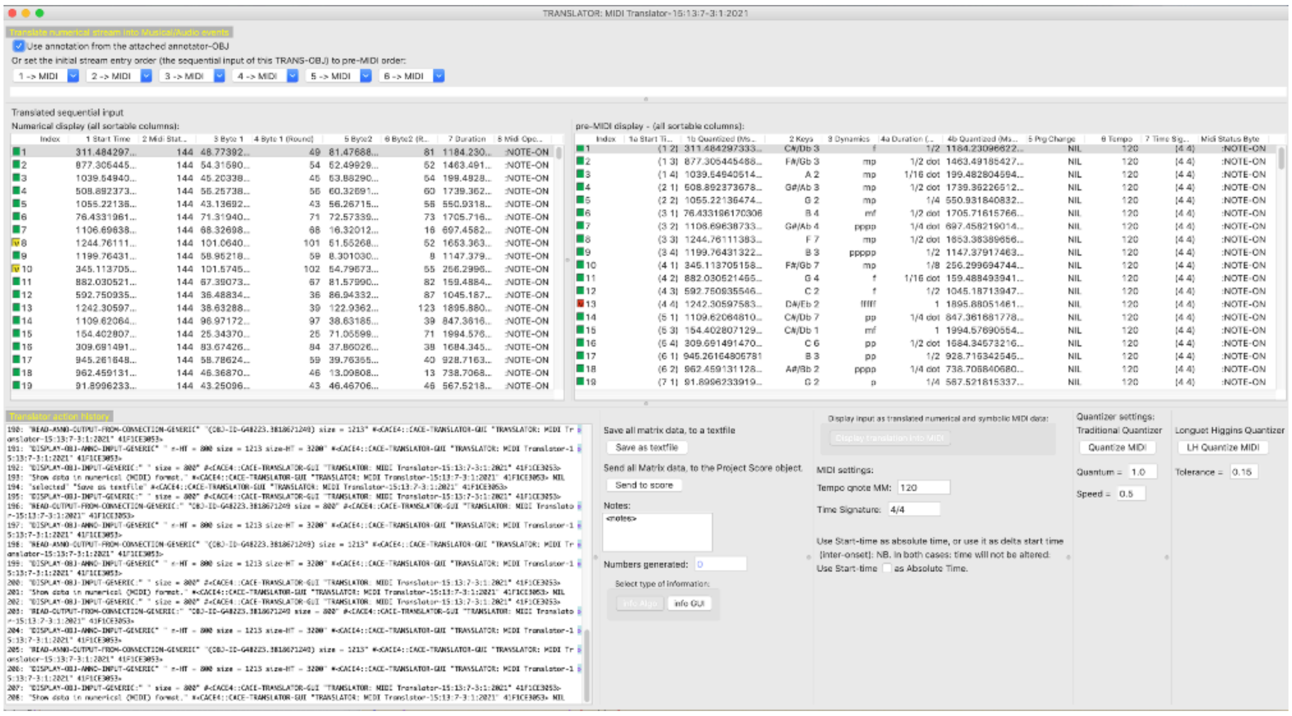
Show input only

Output type of scaling:

Output type of normalization:

Output type of weighting:

Figure 8. The CACE4 Scaler Object with linear scaling values. The list values with a yellow background (right) represent the annotation of the data (left) with a white background.



TRANSLATOR: MIDI Translator-15:13:7-3:1:2021

Translate numerical stream into Musical/Audio events

[✓] Use annotation from the attached annotator-OBJ

Or set the initial stream entry order (the sequential input of this TRANS-OBJ) to pre-MIDI order:

1 -> MIDI | 2 -> MIDI | 3 -> MIDI | 4 -> MIDI | 5 -> MIDI | 6 -> MIDI

Translated sequential input

Numerical display (all sortable columns):

Index	1 Start Time	2 Midi Stat...	3 Byte 1	4 Byte 1 (Round)	5 Byte2	6 Byte2 (R...	7 Duration	8 Midi Opc...
🟩 1	311.484297...	144	48.77392...	49	81.47688...	81	1184.230...	:NOTE-ON
🟩 2	877.305445...	144	54.31590...	54	52.49929...	52	1463.491...	:NOTE-ON
🟩 3	1039.54940...	144	45.20338...	45	53.88290...	54	199.4828...	:NOTE-ON
🟩 4	508.892373...	144	56.25738...	56	60.32691...	60	1739.362...	:NOTE-ON
🟩 5	1055.22136...	144	43.13692...	43	56.26715...	56	550.9318...	:NOTE-ON
🟩 6	76.4331961...	144	71.31940...	71	72.57339...	73	1705.716...	:NOTE-ON
🟩 7	1106.69638...	144	68.32698...	68	16.32012...	16	697.4582...	:NOTE-ON
🟨 8	1244.76111...	144	101.0640...	101	51.55268...	52	1653.363...	:NOTE-ON
🟩 9	1199.76431...	144	58.95218...	59	8.301030...	8	1147.379...	:NOTE-ON
🟨 10	345.113705...	144	101.5745...	102	54.79573...	55	256.2996...	:NOTE-ON
🟩 11	882.030521...	144	67.39073...	67	81.57990...	82	159.4884...	:NOTE-ON
🟩 12	592.750935...	144	36.48834...	36	86.94332...	87	1045.187...	:NOTE-ON
🟩 13	1242.30597...	144	38.63288...	39	122.9362...	123	1895.880...	:NOTE-ON
🟩 14	1109.62064...	144	96.97172...	97	38.63185...	39	847.3616...	:NOTE-ON
🟩 15	154.402807...	144	25.34370...	25	71.05598...	71	1994.576...	:NOTE-ON
🟩 16	309.691491...	144	83.67426...	84	37.86026...	38	1684.345...	:NOTE-ON
🟩 17	945.261648...	144	58.78624...	59	39.76355...	40	928.7163...	:NOTE-ON
🟩 18	962.459131...	144	46.36870...	46	13.09808...	13	738.7068...	:NOTE-ON
🟩 19	91.8996233...	144	43.25096...	43	46.46706...	46	567.5218...	:NOTE-ON

pre-MIDI display - (all sortable columns):

Index	1a Start Ti...	1b Quantized (Ms...	2 Keys	3 Dynamics	4a Duration (...	4b Quantized (Ms...	5 Prg Change	6 Tempo	7 Time Sig...	Midi Status Byte
🟩 1	(1 2)	311.484297333...	C#/Db 3	f	1/2	1184.23096622...	NIL	120	(4 4)	:NOTE-ON
🟩 2	(1 3)	877.305445488...	F#/Gb 3	mp	1/2 dot	1463.49185427...	NIL	120	(4 4)	:NOTE-ON
🟩 3	(1 4)	1039.54940514...	A 2	mp	1/16 dot	199.482804594...	NIL	120	(4 4)	:NOTE-ON
🟩 4	(2 1)	508.892373678...	G#/Ab 3	mp	1/2 dot	1739.36226512...	NIL	120	(4 4)	:NOTE-ON
🟩 5	(2 2)	1055.22136474...	G 2	mp	1/4	550.931840832...	NIL	120	(4 4)	:NOTE-ON
🟩 6	(3 1)	76.433196170306	B 4	mf	1/2 dot	1705.71615766...	NIL	120	(4 4)	:NOTE-ON
🟩 7	(3 2)	1106.69638733...	G#/Ab 4	pppp	1/4 dot	697.458219014...	NIL	120	(4 4)	:NOTE-ON
🟩 8	(3 3)	1244.76111383...	F 7	mp	1/2 dot	1653.38389656...	NIL	120	(4 4)	:NOTE-ON
🟩 9	(3 4)	1199.76431322...	B 3	ppppp	1/2	1147.37917463...	NIL	120	(4 4)	:NOTE-ON
🟩 10	(4 1)	345.113705158...	F#/Gb 7	mp	1/8	256.299694744...	NIL	120	(4 4)	:NOTE-ON
🟩 11	(4 2)	882.030521465...	G 4	f	1/16 dot	159.488493941...	NIL	120	(4 4)	:NOTE-ON
🟩 12	(4 3)	592.750935546...	C 2	f	1/2	1045.18713947...	NIL	120	(4 4)	:NOTE-ON
🟥 13	(4 4)	1242.30597583...	D#/Eb 2	ffff	1	1895.88051461...	NIL	120	(4 4)	:NOTE-ON
🟩 14	(5 1)	1109.62064810...	C#/Db 7	pp	1/4 dot	847.361681778...	NIL	120	(4 4)	:NOTE-ON
🟩 15	(5 3)	154.402807129...	C#/Db 1	mf	1	1994.57690554...	NIL	120	(4 4)	:NOTE-ON
🟩 16	(5 4)	309.691491470...	C 6	pp	1/2 dot	1684.34573216...	NIL	120	(4 4)	:NOTE-ON
🟩 17	(6 1)	945.26164806781	B 3	pp	1/2	928.716342545...	NIL	120	(4 4)	:NOTE-ON
🟩 18	(6 2)	962.459131128...	A#/Bb 2	pppp	1/4 dot	738.705840680...	NIL	120	(4 4)	:NOTE-ON
🟩 19	(7 1)	91.8996233919...	G 2	p	1/4	587.521815337...	NIL	120	(4 4)	:NOTE-ON

Translator action history

```
190: "READ-ANNO-OUTPUT-FROM-CONNECTION-GENERIC" "(OBJ-ID-G48223.3818671249) size = 1213" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Tr
anslator-15:13:7-3:1:2021" 41F1CE3053>
191: "DISPLAY-OBJ-ANNO-INPUT-GENERIC" " n-HT = 800 size = 1213 size-HT = 3200" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-1
5:13:7-3:1:2021" 41F1CE3053>
192: "DISPLAY-OBJ-INPUT-GENERIC:" " size = 800" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-15:13:7-3:1:2021" 41F1CE3053>
193: "Show data in numerical (MIDI) format." #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-15:13:7-3:1:2021" 41F1CE3053> NIL
194: "selected" "Save as textfile" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-15:13:7-3:1:2021" 41F1CE3053>
195: "DISPLAY-OBJ-INPUT-GENERIC:" " size = 800" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-15:13:7-3:1:2021" 41F1CE3053>
196: "READ-OUTPUT-FROM-CONNECTION-GENERIC:" "OBJ-ID-G48223.3818671249 size = 800" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translato
r-15:13:7-3:1:2021" 41F1CE3053>
197: "DISPLAY-OBJ-ANNO-INPUT-GENERIC" " n-HT = 800 size = 1213 size-HT = 3200" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-1
5:13:7-3:1:2021" 41F1CE3053>
198: "READ-ANNO-OUTPUT-FROM-CONNECTION-GENERIC" "(OBJ-ID-G48223.3818671249) size = 1213" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Tr
anslator-15:13:7-3:1:2021" 41F1CE3053>
199: "DISPLAY-OBJ-ANNO-INPUT-GENERIC" " n-HT = 800 size = 1213 size-HT = 3200" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-1
5:13:7-3:1:2021" 41F1CE3053>
200: "DISPLAY-OBJ-INPUT-GENERIC:" " size = 800" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-15:13:7-3:1:2021" 41F1CE3053>
201: "Show data in numerical (MIDI) format." #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-15:13:7-3:1:2021" 41F1CE3053> NIL
202: "DISPLAY-OBJ-INPUT-GENERIC:" " size = 800" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-15:13:7-3:1:2021" 41F1CE3053>
203: "READ-OUTPUT-FROM-CONNECTION-GENERIC:" "OBJ-ID-G48223.3818671249 size = 800" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translato
r-15:13:7-3:1:2021" 41F1CE3053>
204: "DISPLAY-OBJ-ANNO-INPUT-GENERIC" " n-HT = 800 size = 1213 size-HT = 3200" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-1
5:13:7-3:1:2021" 41F1CE3053>
205: "READ-ANNO-OUTPUT-FROM-CONNECTION-GENERIC" "(OBJ-ID-G48223.3818671249) size = 1213" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Tr
anslator-15:13:7-3:1:2021" 41F1CE3053>
206: "DISPLAY-OBJ-ANNO-INPUT-GENERIC" " n-HT = 800 size = 1213 size-HT = 3200" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-1
5:13:7-3:1:2021" 41F1CE3053>
207: "DISPLAY-OBJ-INPUT-GENERIC:" " size = 800" #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-15:13:7-3:1:2021" 41F1CE3053>
208: "Show data in numerical (MIDI) format." #<CACE4::CACE-TRANSLATOR-GUI "TRANSLATOR: MIDI Translator-15:13:7-3:1:2021" 41F1CE3053> NIL
```

Save all matrix data, to a textfile

Send all Matrix data, to the Project Score object.

Notes: <notes>

Numbers generated: 0

Select type of information:

Display input as translated numerical and symbolic MIDI data:

MIDI settings:

Tempo qnote MM: 120

Time Signature: 4/4

Use Start-time as absolute time, or use it as delta start time (inter-onset): NB. In both cases: time will not be altered.

Use Start-time [ ] as Absolute Time.

Quantizer settings:

Traditional Quantizer

Quantum = 1.0

Speed = 0.5

Longuet Higgins Quantizer

Tolerance = 0.15

Figure 9. The CACE4 MIDI Translator Object with translated MIDI values.

Again take a look at figure 4 (pg. 13) for a CACE4 Strategy setup for a small music composition strategy. Note that at the end of our chain of strategy we have to add a CACE4 Translator Object (as presented in fig. 9, pg. 18). This CACE4 Object is used to make a final translation of the data to MIDI/MXML and send it to the CACE4 Score Object or save it as a column of text in a text-file (.txt).

THE WORLD OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING: TOP DOWN VERSUS BOTTOM UP.

Bottom-up and top-down are the two major approaches of quantifying data in the domain of Artificial Intelligence and Machine Learning. These two-complementary processes are both needed to obtain a more thorough valuation of the used data-set. First of all let's have a look at the so-called bottom-up approach.

The world of data and AI or better: the qualification of data without 'a priori'[38] knowledge about the data is the domain where the use of Artificial Neural Networks can be of great help. But remember: they do not measure or obtain knowledge. They only can quantify and categorise data-sets as numbers represented by float or integer values as a vector into groups! Although sometimes not clear how these networks exactly work they can be seen as 'major' cognitive problem solvers, as stated by Daniel Dennit: "The advent of "deep learning" and Bayesian methods has been viewed with mixed emotions by many in cognitive science, Why? The fact that these new cognitive fabrics work so well is astonishing and delightful, and their applications are going to sweep the world, but … although they will give us great answers to hard questions like never before, they won't be able to tell us why." (Dennit 2017, pg. 316).

The other method used is top-down where rule-based knowledge is used in order to classify data. One of the well-known approaches is Experiments in Music Intelligence (EMI) from the American Composer David Cope[39]. You can listen to one of his experiments (Vivaldi) at: https://www.youtube.com/watch?v=2kuY3BrmTfQ See the Literature list (computer music books) as well, with three books by David Cope. CACE4 has besides the Annotator and Translator Objects no top-down approaches of qualifying data at the moment, but plans for implementing such top-down Objects are in the pipeline.

FUTURE DEVELOPMENTS.

Although intentionally not designed for doing DSP, CACE4 at a later point can be incorporated with DSP processes as well. This is very well possible with the use of the Foreign Language Interface (FLI) Library already available in the LispWorks IDE. With the use of this Library blocks of compiled C-code can be incorporated into the CACE4 program. For making use of (analysed) Audio with the use of a Short Time Fourier Transform (STFT); SPEAR files can be used as input next to Text-files and SMF.

---

[38] Without 'a priori'; means without any initial or previous knowledge and is a well-known term of the domain of Philosophy. See: https://en.wikipedia.org/wiki/A_priori_and_a_posteriori

[39] For more information see: http://www.computerhistory.org/atchm/algorithmic-music-david-cope-and-emi/

As previously stated by isolating every process into a separated CACE4 Object with a unique internal design gives the program his extending ability and flexibility for future extension and adaptations[40]. Top-down AI, knowledge based will be implemented as frame-based techniques as suggested by Winston. See: https://www.csee.umbc.edu/courses/771/papers/nebel.html for more about this AI technique.

Further developments will focus on graphics with a true 3D plot extending into multiple dimensional views, with the possibility of rotation and zoom in/out. Further development of the Annotator Object with a MXML implementation and an implementation of the SMF format 1 are also on the todo list.

CONCLUSIONS.

Although CACE4 has its limitations (more different input file formats need to be added to the program), but with a few extra points of attention it can be well used for the tasks it originally was conceived for.

Music Information Retrieval

As it is now (version CACE4 00b.97.x), CACE4 functions with its limitations as expected. The most important limitation is not the way the program works, but the restrictions on the formats it can read and write (IO). Although bugs do exist, they are tracked and will be eliminated in versions to come. - Most of them are not really bugs but are 'lose ends' of software coding which needs extra attention with some extra coding[41]. Therefore, these limitations should be lifted as soon as possible in order to give the CACE4 program the necessary addition to access and use the sources of M(usic)XML as a professional program centred around calculations for score writing. Especially if we want to take further music notation details into account.

As previously stated DSP Algorithms have been omitted in this version of CACE4. But routines as STFT could easily be added to the package of CACE4. For now the CACE4 SPEAR Object does its job adequately the way CACE4 works, but it would add many more opportunities for the analysis possibilities of CACE4.

Education

For educational purposes, it still has to be tested on a larger group of students. Before this can take place

[40] The total size for the data-sets used is only limited by the size of RAM available (in the older versions of LispWorks (a 32bit version); CACE4 therefore is limited to the upper limit of 2 GB of RAM, which can be reached in an intensive data session, when a lot of calculations and data is involved. The latest version of CACE4 is created with LispWorks 8.x the professional (64bit) version. Therefore, in the latest version of CACE4 (64bit version) these limitations are omitted (16 GB of RAM can be addressed).

[41] "Bugs" as they are called, can be split in at least three different types: first of all, the "typo" (or syntactic) error, these can be found by the compiler/interpreter and are rather easy to eliminate. Then we have the "thinko" error: a more difficult one to spot. While not been seen by the compiler/interpreter it is more comparable with a semantic error, or a construction (read design) error in the source code (see for more explanation on these two errors type's: Dennit 2017, pg. 229). A third 'error, but related to the "thinko" can be spotted as a "loose end" error. In first instance a rather good idea of how to construct the code, but by lack of time never really finished: the code as it is, is just not working the way it should be. Easy to notice but hard to get rid of: what was the original idea behind this code? Take a look at https://stackoverflow.com/questions/7849684/what-is-semantic-errors-in-c-language-give-some-examples for addressing errors in the computer language C/C++: syntactic, semantic and design error.

additional information about the algorithms used and the way the GUI is working should be added to all Objects as text-info.[42] For now new text has been added and ART2 makes use of an excerpt of the thesis of the author: “From Music Information Retrieval (MIR) to Information Retrieval for Music (IRM).”(Koenders 2016) will be used as long as they are adequate in covering the topics as they are presented.

Algorithmic composition:
Up to now (summer 2022) several compositions[43] have been created with the use of CACE4. Some were created with initially generated data-sets from fractal, attractor and random calculations and others were generated from data files found on the internet.

In the case of ‘Scope’ a composition for 2 spring drums and 6 ceramic tiles the strategy of "imitation” of generated material (Brownian movements, Random Clouds and Bifurcation) by using the CACE4 MATH-Generator Object STAPS has been used: thus, creating imitations and variations of our generated data-set. In this particular case has proven to be very valuable indeed[44].

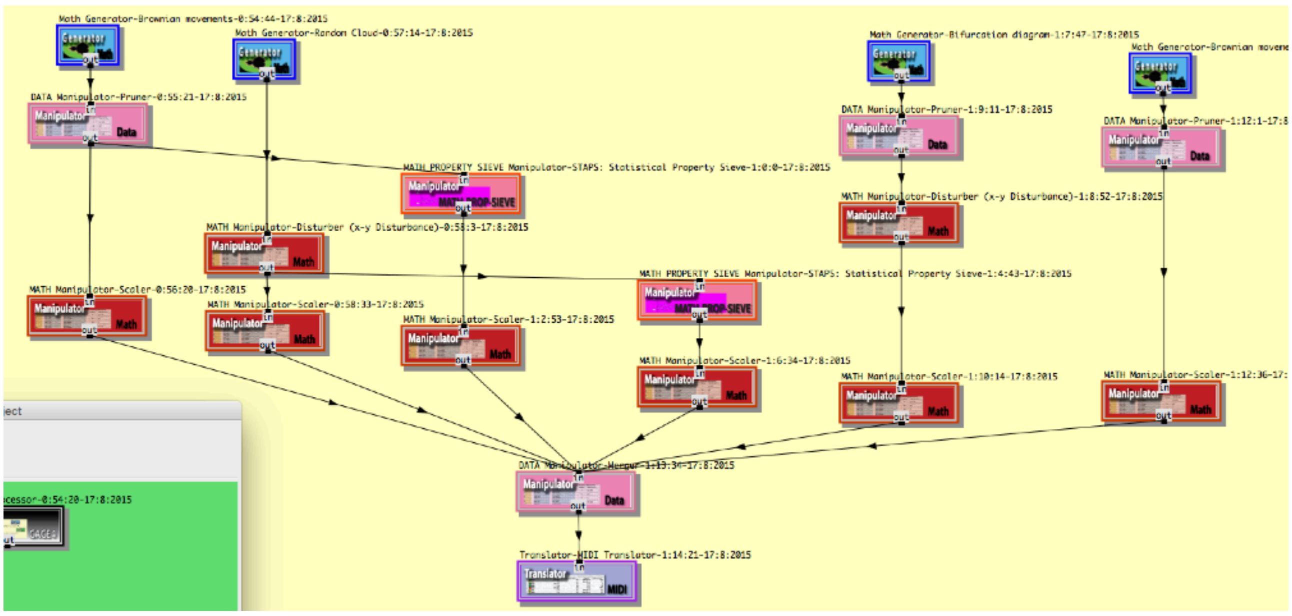

Math Generator-Brownian movements-0:54:44-17:8:2015
Generator
out
Math Generator-Random Cloud-0:57:14-17:8:2015
DATA Manipulator-Pruner-0:55:21-17:8:2015
Manipulator
in
Data
MATH PROPERTY SIEVE Manipulator-STAPS: Statistical Property Sieve-1:0:0-17:8:2015
MATH PROP-SIEVE
MATH Manipulator-Disturber (x-y Disturbance)-0:58:3-17:8:2015
Math
MATH Manipulator-Scaler-0:56:20-17:8:2015
MATH Manipulator-Scaler-0:58:33-17:8:2015
MATH Manipulator-Scaler-1:2:53-17:8:2015
MATH PROPERTY SIEVE Manipulator-STAPS: Statistical Property Sieve-1:4:43-17:8:2015
MATH Manipulator-Scaler-1:6:34-17:8:2015
Math Generator-Bifurcation diagram-1:7:47-17:8:2015
DATA Manipulator-Pruner-1:9:11-17:8:2015
MATH Manipulator-Disturber (x-y Disturbance)-1:8:52-17:8:2015
MATH Manipulator-Scaler-1:10:14-17:8:2015
Math Generator-Brownian moveme
DATA Manipulator-Pruner-1:12:1-17:8
MATH Manipulator-Scaler-1:12:36-17:
DATA Manipulator-Merger-1:13:34-17:8:2015
Translator-MIDI Translator-1:14:21-17:8:2015
Translator
MIDI
ject
cessor-0:54:20-17:8:2015
CACE4

Figure 10. The CACE4 strategy setup for the algorithmic composition Scope. NB This has been made with an earlier version of CACE4 (0d.56.07 - 2016).

[42] Approx. 95 % of all CACE4 Objects have additional information about the GUI and of the algorithms used. The latter one needs some fine tuning.

[43] List of compositions: For small ensemble: ‘Zwicky’s Box’ (2015). And for solo instrument: ‘Argos Pansonos’ (2013) for piano and ‘Scope’ (2016) for spring drums and ceramic tiles. It is also used to create MIDI controller data and notes for the four compositions with Computer Graphics (created in MAYA from Autodesk, see for more information: https://www.autodesk.com.au/products/maya/overview) of the Dutch artist: Willem Willemse (for further information see: http://www.willemwillemse.com).

[44] Listen to some of the compositions created by the Author and take a look at the scores as well at: https://mgm2.home.xs4all.nl/compositions.html or use michelkoenders.com and look for the composition ‘Scope’.

New computer compositions will explore different compositional techniques by creating different setups of our strategy in CACE4. This flexibility offered by CACE4 is one of the more interesting features of the program. As data-sets can be manipulated rather quickly and by having flexibility in creating different strategies the musical output for a composition can be generated rather easily and swift. This offers the user the possibility to generate a lot of composition material in a rather short time. Thus creating many options for experimentation with the musical material needed for the composition[45].

The prepublication of CACE4 version: 0c.97.x can be found at: https://mgm2.home.xs4all.nl/cace4.html

[45] This process can be automated by using the AutoFlow option in the process window. When a CACE4 Object has its AutoFlow option selected only the CACE4 Generator Objects needs to be changed all other CACE4 Objects in the chain are subsequently recalculated.

A LITERATURE LIST.

Computer composition books:

Editors, et al. 1985. Foundations of Computer Music. Cambridge, Massachusetts London, England: The MIT Press. Computer Music.

Ariza, Christopher. 2005. *An Open Design for Computer-Aided Algorithmic Music Composition: athenaCL*. Boca Raton, Florida USA.: Dissertation.com. Music & Informatics.

Barbaud, Pierre. 1965. *Initiation a la composition musicale automatique*: Dunod, Paris 1966. Music & Informatics.

Cope, David. 1991. *Computers and Musical Style*. Oxford, UK: Oxford University Press. Music & Informatics.

Cope, David. 2001. *Virtual Music Computer Synthesis of Musical Style*: Cambridge, Massachusetts London, England: The MIT Press. Music & Informatics.

Cope, David. 2005. *Computer Models of Musical Creativity*. Cambridge, Massachusetts London, England: The MIT Press. Music & Informatics.

Editors, et al. 1985. *Foundations of Computer Music*. Cambridge, Massachusetts London, England: The MIT Press. Computer Music.

Loy, Gareth. 2006. *Musimathics, the mathematical foundations of music, volume 1*. 2 vols. Vol. 1. Cambridge, Massachusetts London, Engeland: The MIT Press. Music, Mathematics and Informatics.

Loy, Gareth. 2007. *Musimathics, the mathematical foundations of music, volume 2*. 2 vols. Vol. 2. Cambridge, Massachusetts London. England: The MIT Press. Music, Mathematics and Informatics.

Moore, F. Richard. 1990. *Elements of computer music*. London: Prentice-Hall. Music and Informatics.

Nierhaus, Gerhard. 2009. *Algorithmic Composition, Paradigms of Automated Music Generation*. Wien (Viena): Springer-Verlag. Algorithmic Composition, Music and Informatics.

Roads, Curtis. 1996. *The computer music tutorial*: Cambridge, Massachusetts London, England: The MIT Press. Computer Music, Music, Mathematics and Informatics.

Taube, Heinrich K. 2004. *Notes from the Metalevel, Introduction to Algorithmic Music Composition*. Edited by Marc Leman. 6 vols., *Studies on New Music Research*. London UK.: Taylor & Francis Group. Music & Informatics.

Temperley, David. 2001. *The Cognition of Basic Musical Structures*. Cambridge, Massachusetts London, England: The MIT Press. Music, Cognition and Statistics.

Temperley, David. 2007. *Music and Probability*. Cambridge, Massachusetts London, England The MIT Press. Music, Cognition and Statistics.

Xenakis, Iannis. 1971. *Formalized Music*. Bloomington, London: Indiana University Press. Music & Informatics.

Information Retrieval books:

Büttcher, Stefan, et al. 2010. *Information Retrieval*
Implementing and Evaluating Search Engines. Cambridge, Massachusetts London, England: The MIT Press. Information Retrieval, Informatics.

O'Neil, Cathy, and Rachel Schutt. 2014. *Doing Data Science. Straight talk from the frontline., Nutshell Handbook*: O'Reilly, USA. Informatics.

Provost, Foster and Tom Fawcett. 2013. *Data Science for Business*. O'Reilly Media Inc. USA, Informatics, Big Data.

Data Mining:

Pang-Ning Tan, Michael Steinbach, Anuj Karpaten and Vipin Kumar. 2020. Introduction to Data Mining. Pearson Education Limited, NY NY USA. Data Mining.

USED LITERATURE

Carpenter, Gila, and Stephen Grossberg. 1987. "ART 2: self-organization of stable category recognition codes for analog input patterns." *Applied Optics* 26 (23).

Dennit, Daniel C. 2017. *From Bacteria to Bach and Back. The Evolution of Minds*. New York, London: W.W. Norton & company. Evolution, Phylosophy and Informatics.

Ghezzi, Carlo, and Mehdi Jazayeri. 1982, 1987. *Programming Language Concepts*. New York, Chichester, Brisbane, Toronto, Singapore: John Wiley & Sons. Informatics.

Keene, Sonya E. 1989. *Object-Oriented Programming in Common Lisp, A Programmer's Guide to CLOS*. USA: Addison-Wesley. Informatics.

Klingbeil, Michael. 2009. "Spear." [Software Application]. http://www.klingbeil.com/spear/.

Koenders, Michael G.M. 2016. *From Music Information Retrieval (MIR) to Information Retrieval for Music (IRM)*. Music, Mathematics and Informatics - Thesis.

Lawless, Jo A., and Molly M. Miller. 1991. *Understanding CLOS, The Common Lisp Object System*: Digital Press. Computer Science.

Rameau, Jean-Philippe. 1722. *Traité de l'harmonie réduite à ses principes naturels*. Dover Books, Dover Publications Inc. New York 1971. Music Theory.

APPENDIX 1 LIST OF ALL OBJECTS AVAILABLE IN CACE4 (0B.97.X).

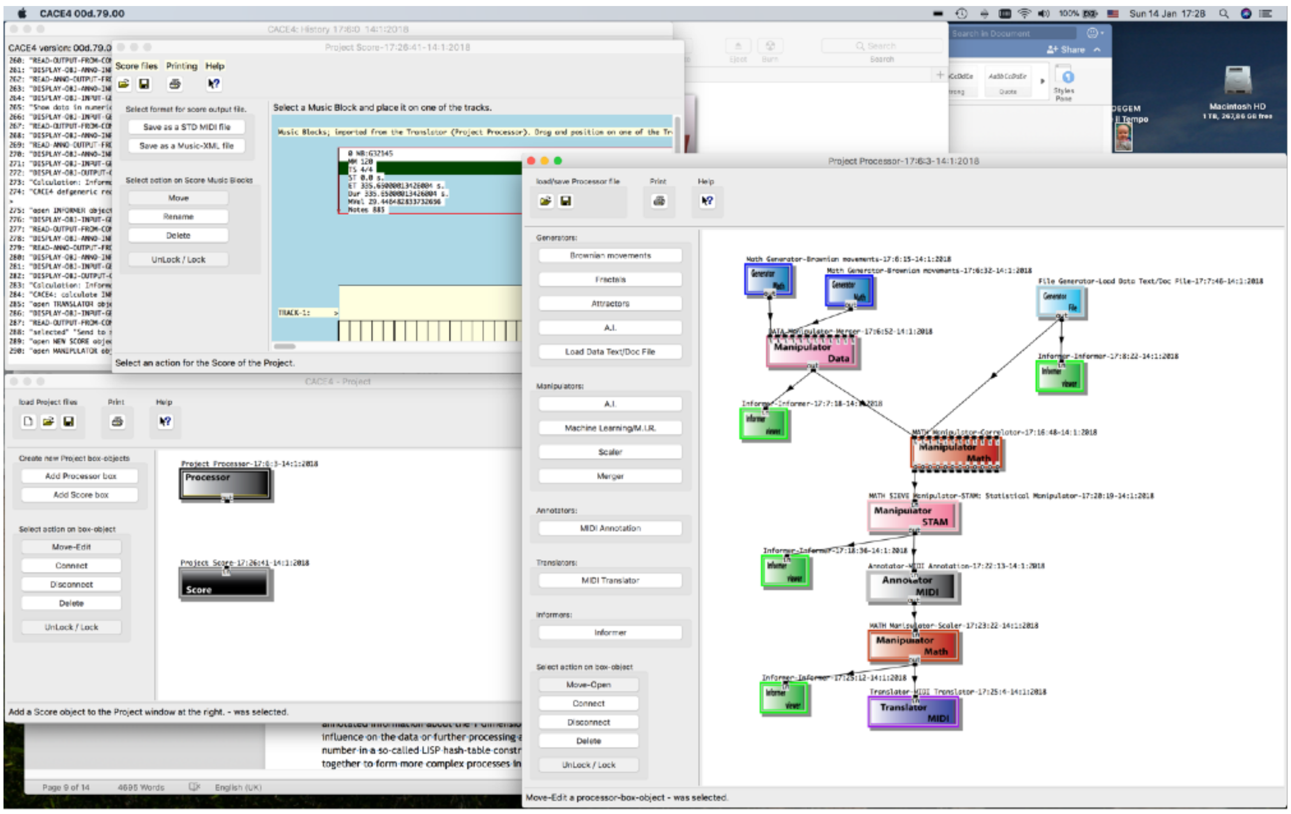
Larger CACE4 Object group	Subclasses:	CAC4 Object description	CACE4 box object colour.	With Annotation
Math Generators	**All fractals:**	Mathematical (functions) Generator Modules for use at the beginning of every strategy as the source of input.	*:blue*	
	Automata (),			-
	Bifurcation (),			-
	Sierpinski (),			-
	Iterated Function System (),			-
	Julia (),			-
	Mandelbrot 1 () & 2,			-
	Mira ().			-
	All attractors:			
	Henon 1 () & 2,			-
	Lorenz () ,			-
	Rössler () ,			-
	Ikeda map ().			-
	All (random) functions:			
	Brownian Movements (),			-
	Random Cloud (),			-
	Chaos on Torus (),			-
	Linear Congruential Method (),			-
	Tendency Masks (),			-
	Number Sequences (),			-
	Function Generator ().			-
File generators	*spear-partials-text-file ()*	Generator Modules for reading files (at the beginning of every strategy) as the source of input.	*:lightblue*	-
	text-file ()		*id.*	-
	midi-file ()		*id.*	-
Artificial Intelligence (AI) generators	*BioArt ()*	A rule based mini world for creating musical material.	:orange	-
Artificial Intelligence (AI) manipulators	*ART2 ()*	Adaptive Resonance Theory 2 Neural Network.	*:orange*	-
	rsm-fuzzy ()	A Fuzzy Logic Controller.		no

DATA manipulators	*pruning ()*	For manipulating the data by deleting, merging, scaling, sorting, splitting, filtering and checking odd-even.	*:palevioletred*	yes
	merging ()			yes
	odd/even ()		*id.*	yes
	scaling ()		*id.*	yes
	shuffler ()	Randomize order.	*id.*	yes
	sorting ()	Sort and stable-sort.		yes
	splitting ()	Separate streams.		yes
	filtering ()	Shelf- and Savitzky-Golay filter.		yes
MATH Manipulators	*math-clusterer ()*	Several techniques of manipulating the data by means of applying different mathematical functions.	*:orangered2*	yes
	calculator ()		*:orangered3*	yes
	correlator ()		*id.*	yes
	disturbance ()		*id.*	yes
	interpolator ()		*id.*	yes
	set-theory ()		*id.*	yes
	math-sieve ()		*:pink*	yes
	math-property-sieve ()		*:orangered1*	no
Machine Learning and Information Retrieval Manipulators	*EM ()*	Expectation Maximization, *k*-means(++) and k-NN (Nearest Neighbour) as Hierarchical Cluster detection Techniques.	*:red*	-
	k-means () & k-means ()++		*id.*	-
	k-NN ().		*id.*	-
	HCA ()	HCT + WPGMA.	*id.*	-
	Recommendation System ()	Linear Regression Techniques.	*id.*	-
Score	No subClasses	The CACE4 for creating a simple score.	*:grey*	-
CACE Processor	No subClasses	The main CACE4 Processor object were all strategies should be developed.	*:black*	-
CACE Project	No subClasses	The CACE4 Project with at least one Processor Object (which contains the strategy) and one Score Object.	No Box	-
Annotator	*midiannotator ()*	The CACE4 Annotator Object for (MIDI) annotating of the data.	*:grey*	yes
	mxmlannotator ()	The CACE4 Annotator Object for (MusicXML) annotating of the data.		yes

Translator	*miditranslator ()* *mxmltranslator ()*	The CACE4 Translator Object for translating the (previously as MIDI annotated) data to the CACE4 Score Object. The CACE4 Translator Object for translating the (previously as MXML annotated) data to the CACE4 Score Object.	*:purple*	yes yes
Informer/Viewer	*informer ()*	The CACE4 Object for Viewing all data in a non-destructive way.	*:green*	yes

Table 2. Table of all 55 CACE4 Objects available in version CACE4 0b.97.x
NB. The objects with no for the 'with annotation column will have annotation available in one of the future updates. The Objects with '-' are omitted of having any annotation information.

APPENDIX 2 CACE4 SESSION AT WORK.



CACE4 00d.79.00
CACE4: History 17:6:0 14:1:2018
CACE4 version: 00d.79.0
Project Score-17:26:41-14:1:2018
Score files
Printing
Help
Select format for score output file.
Save as a STD MIDI file
Save as a Music-XML file
Select action on Score Music Blocks
Move
Rename
Delete
UnLock / Lock
Select a Music Block and place it on one of the tracks.
TRACK-1:
CACE4 - Project
load Project files
Print
Help
Create new Project box-objects
Add Processor box
Add Score box
Select action on box-object
Move-Edit
Connect
Disconnect
Delete
UnLock / Lock
Project Processor-17:6:3-14:1:2018
Processor
Project Score-17:26:41-14:1:2018
Score
Project Processor-17:6:3-14:1:2018
load/save Processor file
Generators:
Brownian movements
Fractals
Attractors
A.I.
Load Data Text/Doc File
Manipulators:
A.I.
Machine Learning/M.I.R.
Scaler
Merger
Annotators:
MIDI Annotation
Translators:
MIDI Translator
Informers:
Informer
Select action on box-object
Move-Open
Connect
Disconnect
Delete
UnLock / Lock
Math Generator-Brownian movements-17:6:15-14:1:2018
Math Generator-Brownian movements-17:6:32-14:1:2018
File Generator-Load Data Text/Doc File-17:7:46-14:1:2018
DATA Manipulator Merger-17:6:52-14:1:2018
Informer-Informer-17:8:22-14:1:2018
Informer-Informer-17:7:18-14:1:2018
MATH Manipulator-Correlator-17:16:48-14:1:2018
MATH SIEVE Manipulator-STAM: Statistical Manipulator-17:20:19-14:1:2018
Informer-Informer-17:18:36-14:1:2018
Annotator-MIDI Annotation-17:22:13-14:1:2018
MATH Manipulator-Scaler-17:23:22-14:1:2018
Informer-Informer-17:25:12-14:1:2018
Translator-MIDI Translator-17:25:6-14:1:2018
Generator
Math
File
Manipulator
Data
Informer
viewer
STAM
Annotator
MIDI
Translator
Sun 14 Jan 17:28
Macintosh HD

```
260: "READ-OUTPUT-FROM-CO
261: "DESPLAY-OBJ-ANNO-IN
262: "READ-ANNO-OUTPUT-FR
263: "DESPLAY-OBJ-ANNO-IN
264: "DESPLAY-OBJ-INPUT-G
265: "Show data in numeric
266: "DESPLAY-OBJ-INPUT-G
267: "READ-OUTPUT-FROM-CO
268: "DESPLAY-OBJ-ANNO-IN
269: "READ-ANNO-OUTPUT-FR
270: "DESPLAY-OBJ-ANNO-IN
271: "DESPLAY-OBJ-INPUT-G
272: "DESPLAY-OBJ-OUTPUT-
273: "Calculation: Inform
274: "CACE4 defgeneric re
>
275: "open INFORMER object
276: "DESPLAY-OBJ-INPUT-G
277: "READ-OUTPUT-FROM-CO
278: "DESPLAY-OBJ-ANNO-IN
279: "READ-ANNO-OUTPUT-FR
280: "DESPLAY-OBJ-ANNO-IN
281: "DESPLAY-OBJ-INPUT-G
282: "DESPLAY-OBJ-OUTPUT-
283: "Calculation: Inform
284: "CACE4: calculate IN
285: "open TRANSLATOR obje
286: "DESPLAY-OBJ-INPUT-G
287: "READ-OUTPUT-FROM-CO
288: "selected" "Send to 
289: "open NEW SCORE obje
290: "open MANIPULATOR ob
```

```
Music Blocks; imported from the Translator (Project Processor). Drag and position on one of the Tr
0 NR:632545
MM 120
TS 4/4
ST 0.0 s.
ET 335.65000013426004 s.
Dur 335.65000013426004 s.
MVel 29.448482833732656
Notes 885
```

Select an action for the Score of the Project.

Add a Score object to the Project window at the right. - was selected.

Move-Edit a processor-box-object - was selected.

Figure 11. A typical CACE4 session of several open windows. The setup shows a Processor window, a Project window and a CACE4 Score Object.

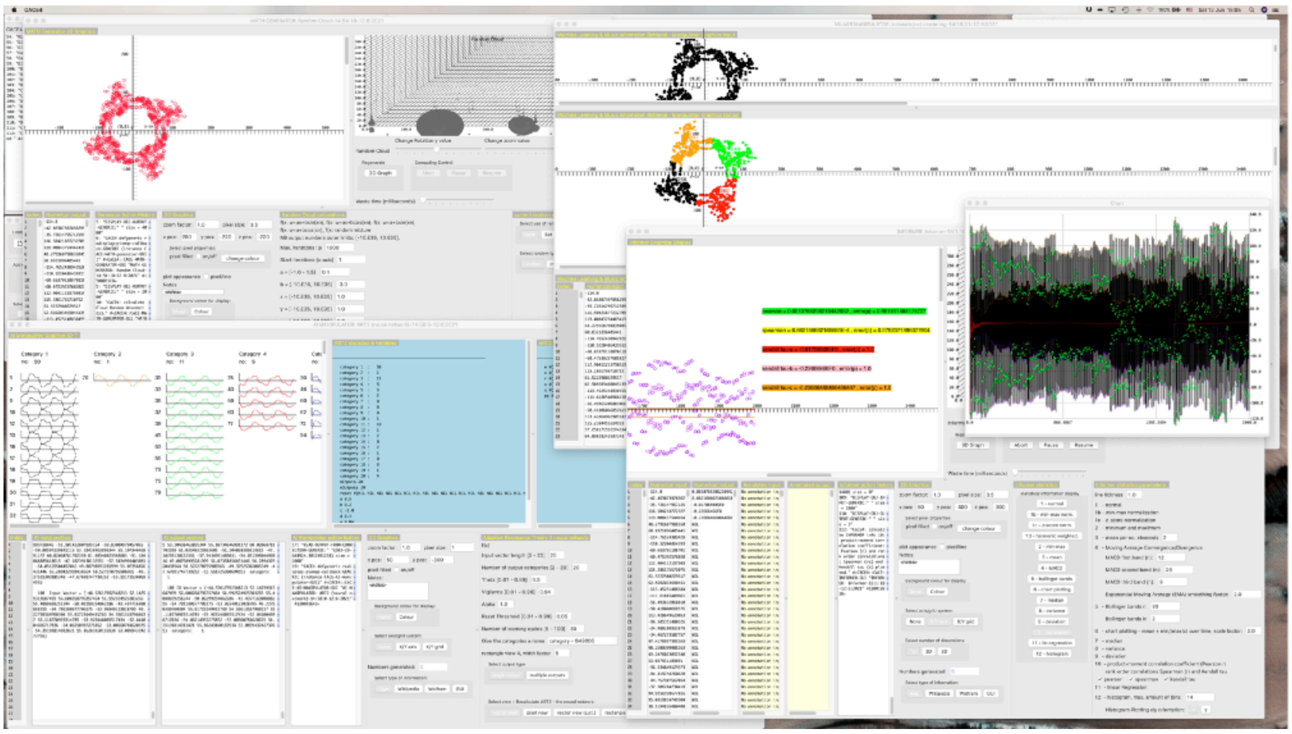


Figure 12. A Processor Object displaying several AI (ART2 NN) and HCT CACE4 Objects.