Using open source music software to teach live electronics in pre-college music education

Hans Roels
University College Ghent - Faculty of Music
Hoogpoort 64
B-9000 Ghent, Belgium
hans.roels@hogent.be

Abstract

A basic course of live electronics is needed in pre-college music education to teach children how to perform on a digital musical instrument. This paper describes the basic components of such a live electronics course, examines whether open source music software is suited to realize these components and finally presents Abunch, a library in Pure Data created by the author, as a solution for the potential educational disadvantages of open source music software.

Keywords

live electronics, music education, Pure Data, digital musical instruments, open source music software

1 Introduction

Since more than a decade home computers and laptops have become powerful enough to process sound in real time. This in fact transformed computers into musical instruments. As this change was taking place, computers became more widely available in households and schools. Anno 2010 the computer has probably become the most widespread musical instrument in a large part of the developed world.

This unique situation urgently prompts us to rethink and redesign our music education which is still largely built upon our traditional knowledge of acoustical instruments and music and to question why pre-college[1] music education in Europe lacks a course of live electronics[2] in which children, teenagers and amateur musicians can learn to perform on a digital musical instrument.

This paper describes the basic components of live electronics courses on a pre-college level, examines whether open source music software -and in specific Pure Data- is suited to realize these components and finally presents Abunch, a library in Pure Data created by the author, as a solution for the potential educational disadvantages of open source music software.

2 Live electronics

2.1 Digital Musical Instruments

It is important to know which fundamental differences exist between live electronic and acoustical music before the content and methodology of a live electronics course can be discussed.

First, in a digital musical instrument (DMI) the sound production unit and the user interface can be seperated and recombined[1]. An acoustical link between both parts as in an acoustical instrument is unexisting and unnecessary. The unique modular nature of a DMI should be central to live electronics education as it is this particular feature that distinguishes it from other traditional instruments.

Secondly, we should ask ourselves what 'performing well' means in a live electronic context as we wish to learn our children and students to play this instrument well. Virtuosity in the digital musical era has an off-stage component which is almost as important as the on-stage one and which is not only different but also more diverse and extensive than in acoustical virtuosity. Developing, building, modifying or adapting a digital instrument is highly important to produce a convincing and expressive performance apart from the classical training.

[1]The term 'pre-college music education' is used to denote all kinds of music education for children, teenagers and grown-ups who haven't taken music courses on a college, university or professional level

[2]Live electronics is used in a broad sense to describe a performance with at least a human performer and an electronic device producing or processing sound

The third difference can be found in the information stream between composers and performers. In electronic music the number of parameters that can be manipulated (before or during a performance) turn a traditional score into a restricted medium to communicate a message between a composer and a performer. Because not all parameters can be notated (in detail) a performer often needs to improvise along more general guidelines. If a performer wants to learn from a composer or from other performers how to perform he has to attend a performance, talk directly to a collegue or composer or consult other media (recordings, texts, source codes, patches, websites, mailing lists, videos,...) than a score. Information has become multimodal in live electronics.

Live electronic music is clearly different from acoustical music, the categories of human activities within music making reflect these changes. The boundaries between composer, improviser, performer and instrument-builder have been blurred and 'performing' has in fact become quite an inadequate term. In general whenever this term is used in live electronics, a larger amount of composition, improvisation and instrument-building is implied than in acoustical music because of the reasons mentioned above. Therefor creativity and autonomy- have become more outspoken in the live electronic music scene.

2.2 Content

Taking into account the specific character of the musicianship and the instrument in live electronics, a basic content of a beginners course for live electronic music is presented. In short the following knowledge domains should be part of this content:

1. Digital Signal Processing techniques
2. Basic audio hardware
3. Mapping techniques
4. History of electronic music
5. Auditory training
6. Sound organisation in real time
7. Performance training

Each of these seven domains is very extensive and could be the subject of a new course. In a beginners course of live electronics these subjects need to be treated only very basically and a selection within each domain has to be made. The seperate implementation in music education of six of these domains is not new and has been researched and applied in classrooms before[2][3]. Introducing mapping techniques on a pre-college level is new and necessary because they are essential to use the full and unique potential of a DMI (see 2.1). The essential parts of these mapping consist of:

1. basic math
2. basic boolean operators
3. comparison operators
4. assignment operator
5. relay switch
6. a module or system to order all this logic and math in time

Because mapping is in fact programming, the components could be a lot more extensive but this selection provides a basic set with a lot of possibilities to connect user interface and DSP in many different ways.

2.3 Methodology

In a live electronics course the sound experience and performance should form the basis of the teaching methodology. Performance requires fast skills and automatisms of the human body. These are in fact feedback loops between the sensoritory information and body gestures. Our brain receives perceptions from our ears, eyes, hands, etc., processes these in a conscious and inconscious way and creates intentions that are embodied in body gestures which adapt and change the sound [4]. This -almost simultaneous- cycle of perception, cognition, intention and movement is action-based and all the separate units are related to the central act of performance. In an action-based methodology the theoretical knowledge can be integrated in listening and performance experiences and foster further progress in sound imagination, experimentation and performance. Music theory and technical knowledge thus are a tool to develop performance -and in general musical- skills.

The forementioned mapping techniques might seem to be boring or very theoretical in a music course, but, if they are integrated in performance and instrument design tasks, they can be fun. One can *hear* whether the result of a mapping logic is right or wrong and consequently recognize a problem and try to solve it... (a method that math or programming teachers would certainly find very attractive).

The multimodal information, the lack of detailed scores and the modular nature of a DMI in live electronics (see 2.1) require a high level of creative input from the children and a need to rely on direct aural information (and not on a score). The teaching methodology in live electronics should therefore be mainly auditory based and encourage personal autonomy and creativity. For example, as there are no fixed and absolute rules about the right coupling of user interface and DSP, it is very important that children have sufficient time and freedom to experiment with those techniques in order to learn more about the different factors (available equipment, physical skills, artistic demands,...) on which this coupling choice depends. These experiments also help to hear to what extent the user interface and mapping define the audio result, even when the same DSP techniques are used.

To summarize, a live electronics course should center on the immediate perception and performance of sound, use background theory and auditory training to improve and develop this musical activity, promote the tools to take advantage of the modular nature of a DMI and foster the creativity and autonomy of each pupil. Any software to learn live electronics should fit in this general framework.

3 Open Source Music Software in live electronics education

Is open source music software suited to teach and learn live electronics? In the next section Pure Data (Pd) is used as an example to answer this question.

3.1 A continuous learning environment

Anyone who wants to learn to play an instrument should regularly have access to his instrument. This is an almost self-evident truth in instrument training. The simple but very powerful argument in favor of open source music software for live electronics education is its accessibility, low cost and ability to run on several operating systems. These features enable schools and pupils to install and use this software at school and at home. In this way pupils can have regular access to their digital musical instrument and can start developing all the subtile automatisms and gestures that are required to perform music on an instrument. As in acoustical instrument training, the main part can be done at home while the class room only serves to guide this continuous process. In this way open source music software enforces the importance of performance in live electronics.

3.2 The combination of user software and programming language

At the moment there is a wide choice of open source software for live electronic music (Pure Data, ChucK, CSound, SuperCollider, etc.). All these programs are in fact combinations of user software and audio programming languages and some pedagogical problems -especially on a pre-college level- may arise because of this combination.[3]

First, a beginner can easily get lost in the massive and confusing range of possibilities and lose his motivation to learn more about electronic music.

Second, if a newbie wants to find information (articles, websites, mailing lists, books,...) about open source software for live electronics, it is often quite technical and requires a lot of inside knowledge. For a beginner some texts or mailing lists look like cryptograms. Although a lot of patches and example files for beginners do exist in a program like Pure Data, the level is often too high for pre-college teenagers, especially for the ones that have no background in electronic music or software programming.

Third, as I started teaching live electronics with Pd to teenagers in a music school, I noticed that there is another disadvantage to this combination: for a musician who wants to perform or compose it takes too long before he can be creative with the instrument design. He has to know too many basic units and syntax rules before he can produce sound and start combining them.

Of course the combination of user software and programming language has a great pedagogical advantage, especially in a graphical environment like Pd where there is no compiler and no split between the source code and the compiled program. This kind of transparancy helps a tutor to deal with more theoretical knowledge within a performance context and corresponds very well to the action-based methodology of live electronics courses. At the same time this transparancy also helps to abstract or transcend the software that is used in the classroom and to learn more about electronic music and its performance in general. By seeing the basic source code and learning more about fundamental techniques, it becomes easier to

[3]It is no coincidence that most software packages for live electronic music are more or less programming languages. This reflects the fundamental changes in Digital Musical Instruments and in the performance of live electronic music described in 2.1

recognize similar procedures in other software for live electronic music.

Finally as programming languages these forementioned open source music software packages have all the basic tools for learning and applying the mapping techniques. Understanding and using the modular nature of a DMI becomes feasible in a live electronics course using this kind of software.

4 Abunch

4.1 Content

Abunch[4] is a collection of 60 high-level objects (so-called abstractions) in Pd and an additional set of information files. The aim of the main part of the abstractions is to perform electronic music while the remaining abstractions and the info files demonstrate, analyse or explain techniques and musical applications in live electronic music.

The abstractions provide a set of ready made objects to

- record and play sound files (from hard disk and memory)
- manipulate and process sound (effects)
- generate sounds (synthesizers)
- prepare control data (sequencers with different graphical interface)
- synchronize control data (clocks)
- analyse sound and control data (oscilloscope, spectrum analyzer,...)
- record control data to a score
- receive data from common interfaces
- algorithmically generate control data

These Abunch objects use techniques like FM synthesis, granular synthesis and random walk algorithms. The majority of the abstractions were made by the author while approximately one fourth is based on files by other authors.[5] All Abunch objects share a common architecture and can easily be connected with each other (and with native Pd objects) to create all kinds of custom made live electronics. Thus in the first place this library is an active toolbox to experience and learn electronic music.

To start off with performing, Abunch also contains a lot of information and documented patches. Every Abunch object has a help file that explains its workings and that is accessed using the normal help procedure in Pd -right-clicking an object-. Moreover there are more than 40 example files that not only demonstrate the general application rules of Abunch objects but also the internal structure of general audio techniques (FM synthesizer, loopstation device,...) and general musical 'recipes'. The latter uses guidelines and tricks -for the musical application of specific techniques- that have been developed in the last 60 years by composers and performers in electronic music of different styles.

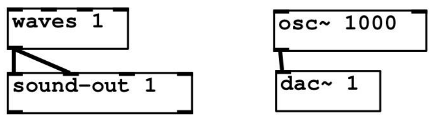
Finally a 'Quick Start' tutorial was made about Abunch and a mapping tutorial. This last tutorial gives some simple examples on how to connect user interface and sound production using the basic operators explained in 2.2.

4.2 Simplified procedures

In the first place the ease of use for beginners was obtained by providing high-level objects as building blocks but the installation and working procedure was also simplified as much as possible.

The installation is straightforward and only requires 1. the Pd core version (called 'Vanilla') which implies that no externals need to be installed and 2. the Abunch folder with files.

If you want to create a new object in Pd, you have to know the name, use a '~' sign to discriminate audio from control objects and provide a set of arguments which refer to specific parameters and functions of that object. In Abunch you can start creating new objects by typing the name (not caring about the '~' in the name) and an unique number as an argument. Thus only one type of argument is used (to enable a general preset system)[6].



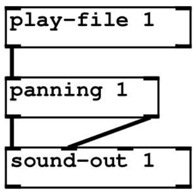
The different procedures in Abunch (left) and Pure Data (right). In Abunch the argument (the number after the object name) refers to the preset system. In Pure Data there are more options: '1000' refers to the frequency of the oscillator 'osc~' object and '1' to one audio hardware output of the digital-to-analog converter (dac~).

[4]Abunch is available for download at www.hansroels.be/abunch.htm and is released under the Creative Commons GNU General Public Licence.

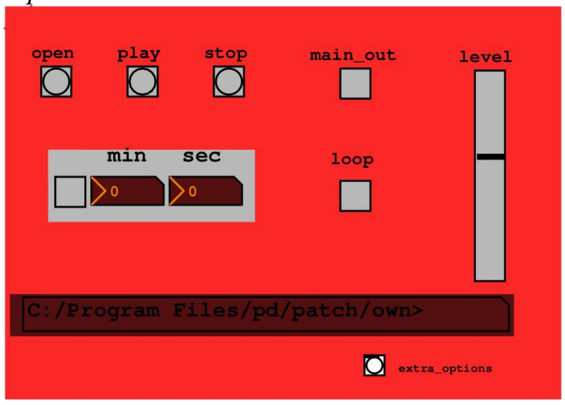
[5]Authors such as Miller Puckette, Frank Barknecht and Tristan Chambers

[6]Giving every Abunch object an unique argument enables to store several instances of the same object in the presets.

Once a new object is created, one can start connecting objects. In Pd objects can receive numbers, audio signals, lists or all kinds of messages for special functions. In Abunch the connection types were reduced to numbers (for control data), audio signals and 2 special connection types: a 'clock' signal to synchronize time related objects and a 'record' connection to combine record and play objects into live recording units.



Three Abunch objects connected to each other, the control window of every object can be closed or opened.



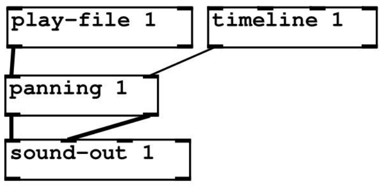
The control window of the 'play-file' object.

4.3 Goal

The main goal of Abunch is to provide a practical open source software tool that enables beginners to learn more about the musical possibilities in real time of a computer. Users can listen to basic tools and techniques and actively learn more about these techniques. In this way Abunch fosters an active teaching and learning method in which sound is the main focus and theory can be integrated in the performance. Pupils can immediately start experimenting with computer sound by combining these objects and techniques with each other to create their own desired sound devices and thus their own set of knowledge. An experimental attitude and a critical, personal opinion and methodology is encouraged by the open-ended architecture of Abunch [5][6]. The analysing objects in the Abunch library enable the students to test and evaluate the other objects and their own built patches and thus help them to take control of their own learning.

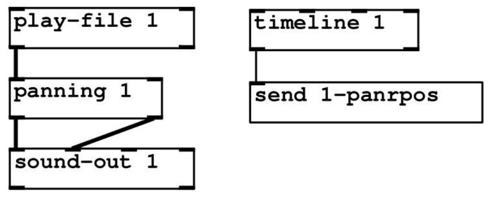
4.4 Advanced features and integration within Pure Data

Abunch was launched in 2008 and tested and used with a large number of children and students. As they got acquainted with the library and its way of working I started noticing that for some of them Abunch was becoming too easy and restricted and a mode to combine user friendliness and more advanced features had to be found. One of the solutions was to hide the more advanced procedures and use the 'wireless' sends and receives in Pd. Almost all graphical user interface (GUI) objects (faders, knobs, switches,...) within Abunch objects can be easily controlled by other objects via the inlets, the usage of the GUI elements can thus be automatized. For ease the control values are normalized to a range of 0 to 127. This also facilitates the use of MIDI hardware to control Abunch objects. The restrictions of this easy MIDI-like procedure can be superseded by using the sends and receives system. Every Abunch object can print out a list of hidden send and receive names to which values within any range can be sent. Via this 'hidden' procedure more parameters can be controlled and adjusted. In a similar way more advanced features were added without changing the easy-to-use layout.[7]



Normal procedure in Abunch: a sequencer object called 'timeline' controls the 'pan' fader within the 'panning' object by connecting it to the right input. This input of the 'panning' object is normalized to the range 0 - 127 just like the output from 'timeline'.

[7]Extra features were also added because Abunch was used by the author in his PhD-research.



Advanced procedure: the range of the values in 'timeline' can be adjusted in the 'extra_options' of this object and are sent to the 'send name' 1-panrpos of the 'pan' fader in the 'panning' object.

In a further stage pupils and students can start using native Pd objects to add more possibilities to Abunch. One part of the example files demonstrates how these native Pd objects can be combined with Abunch objects. The procedures in Abunch are made as similar as possible to Pd procedures to aid the transition from Abunch to Pd. In general Abunch uses a reduced number of procedures, thus the main challenge for the transition is to learn new methods and possibilities. Two specific changes were added to Abunch though (and these were mentioned before in 4.2): the '~' is not used in the name of audio objects and the argument of an object only refers to the general preset system.

4.5 Future plans

Abunch is of course a work in progress with room for improvement, especially because until today it is a solo project and the pace of development is mostly dictated by short-term educational demands.

A frequently heard criticism from pupils is the simple layout. Another problem is the large number of files that is needed to use abstractions and presets within a main file. There is no direct method to bundle all the files in one folder[8] and only a workaround solution was found. This problem is pedagogically relevant because pupils perform and pratice at home and often forget to copy a number of files when they return to the classroom.

Future versions of Abunch will hopefully include the following:

- more example files (about the musical application of techniques)
- more usage of the data structures in Pd to create a more attractive and diverse layout
- a neat and uniform structure within each object with information and comment in the source code
- a style guide for other developers that want to make new Abunch (or similar) objects
- an easy-to-use template for composition and improvisation algorithms

[8]There is no procedure or object in the core version of Pd ('Vanilla') to know the current file name, to copy files and to create a new folder.

5 Conclusion

The modular nature of a Digital Musical Instrument and thus mapping techniques are considered as central parts of any live electronics course in pre-college education. Open source music software for live electronic music is well adapted to teach these mapping techniques and -because of its low cost and accessibility- ensures the tutor and student regular access to their musical instrument which is a prerequisite for any course of instrument training. Therefor it is possible to use open source music software like Pd successfully to teach children to perform live electronic music.

A solution for the massive amount of possibilities and the insufficient user friendliness in an audio programming language like Pd is found in the development of a library of high level objects like Abunch. This library is a balanced mixture of performance and theory-orientated objects with simplified ready-to-use procedures and more advanced hidden features.

References

[1]Miranda, Eduardo Reck, and Marcelo M. Wanderley. *New Digital Musical Instruments: Control And Interaction Beyond the Keyboard*. Pap/Com. A-R Editions, 2006.

[2]Brown, Andrew. *Computers in Music Education: Amplifying Musicality*. Routledge, 2007.

[3]Landy, Leigh. *The ElectroAcoustic Resource Site (EARS).* Journal of Music, Technology and Education, no. 1 (November 2007): 69-81.

[4]Leman, Marc. *Embodied Music Cognition and Mediation Technology*. 1st ed. MIT Press, 2007.

[5]Holland, Simon. *Artificial Intelligence in Music Education: A Critical Review*. In Readings in Music and Artificial Intelligence, Miranda, Eduardo Reck, ed. Routledge, 2000. .

[6]Smith, Brian. *Artificial Intelligence and Music Education.* In Readings in Music and Artificial

Intelligence, Miranda, Eduardo Reck, ed. Routledge, 2000.