Open, High and Low: Getting Deep in the Sound Source Code

Hernani Villaseñor Ramírez
Universidad Nacional Autónoma de México
Posgrado en Música
Xicoténcatl 126, Mexico City, 04100, Mexico
hernani.vr@gmail.com

This text presents a research progress about the relationship between artistic practice and technological development in the field of sound and computer music made with code. In this context, the materials of sound artists and computer musicians are the textual code and the sound generated or controlled by the code. The discussion is related to an apparent dissociation between artistic practice and technological development. To understand this relationship I start from the computer science’s concept of high and low programming levels. Based on this, I observe that artistic practice is usually centred in the high level while technological development points out to both levels. To study this relationship I designed a graph in which artistic practice and technological development have the categories of layers and are connected to the high and low levels by the source code. The access to lower levels makes it possible to understand the control structures that define modes of artistic production of the software, which implies somehow different discourses and questions. This is possible when the source code of software is open, but in order to understand the technological structures in lower levels, from an artistic perspective, a specialized knowledge is required.

1. INTRODUCTION

Computer music is made in many ways, one of them is by writing source code with programming languages, in this case, artistic creation and technological development find a common place in writing source code. The open source programming languages, besides allowing to program computers to create music, show their structures and enable the development of new objects, classes and libraries. The relationship between art and technology can be studied in this context, and although it is not a new topic in computer art it continues to raise questions as artistic practices are transformed, new technologies emerge and exiting ones are updated.

This text articulates the concepts of source code, programming languages, openness, and high and low levels in order to try to understand the relationship between art and technology in the field of computer music creation made with source code from the perspective of the artistic practice. For this, I propose a conceptual research space divided into layers representing artistic creation and technological development, which are contained within the concept of source code and delimited by the low and high levels of programming languages.

2. CODE, LANGUAGES AND LEVELS

The source code consists of symbols, words and characters organized in a text that a human writes to instruct a computer to perform actions. Source code writing is carried out under the rules of programming languages, that, according to Daniel Temkin (2017), are formal languages with precise syntax and semantics which serve to express commands to a computer. Programming languages can be classified in high level by the proximity to human understanding, and in low level by the proximity to the computer.

Bolanakis et al. (2011) say that “a computer program written in a high-level language uses statements which are built from a set of –familiar to human– symbols […] that act as a link between the humans’ reflection and the machine tasks” (p. 526). In addition, these languages present a high degree of abstraction that makes them flexible to program, which facilitates the expression of ideas. For this reason, the source code and algorithms to generate sound are usually written with high-level languages, for example, Python or SuperCollider.

On the other hand, Marta Vazquez-Vilar et al. (2018) mention that low-level programming languages are located just above the machine.
code, provide low abstraction and have a close relationship with the machine since they use syntactic elements that are very similar to the code executed by the computer hardware. For this reason, low-level programming languages are generally used to program the computer’s processors, for example, assembly languages and C. Although David Chisnall (2018) argues that C can no longer be classified as low level for several reasons that don’t allow a clear understanding of what happens at a lower level when it is programmed in this language.

In both cases, high and low level, the source code is written by humans, so the computer needs to translate it into machine code through a compilation or interpretation process to execute it and produce a result, either a sound, image or some other type of output.

Figure 1: High and low levels need to be translated.

This article uses the high and low level classification to situate artistic practice and technological development within musical creation and programming. The goal is to observe the intertwined processes that occur in that relationship. From my perspective, artistic creation made with programming languages occurs above many technological layers that can be accessed when the code of its structure is open. Seen from this perspective, the questions are: What happens when artistic creation is carried out at a level supposedly delegated to technological development? How does artistic practice builds its relationship with technological development?

3. OPEN

The word open in this text refers to showing how the parts of a program are made and sharing the technological and artistic processes of an artistic creation practice. David Berry (2011) proposes the critical code concept to describe the “code that is written to open up existing closed forms of proprietary computer code” (p. 53), allowing users to read and modify it. With this, Berry refers to free software and open source projects and suggests that a critical code requirement is that the source must be available to observe its processes.

For his part, Hernando Barragán (2007) states that software is art, he argues that by taking the source code out of its computational context we can see it as artistic material and appreciate it as an aesthetic and formal object. Barragán claims an open code behind computer art and urges a commitment of experimentation that reflects contemporary thinking through code in computational art.

According to Marije Baalman (2015), in practices such as live coding, some practitioners build bridges between the artistic and the technological when they write and share their own tools. The author suggests a constant collaboration between disciplines, creators and users. In this context, the open source code allows access to the technological and artistic thinking embodied in the code.

Under these perspectives, computer art has the possibility to show the material, the processes and the ideas implicit in it, in this case, from the source code of the artistic pieces and technological objects, as in the case of the artistic practice of live coding, in which the process of programming sound, image or choreography is shown by projecting the code written during a performance to the audience (Lee 2019); another example are the programs that have their source code accessible in a repository.

However, this openness goes hand in hand with different technological knowledge, first, to use a software, and then, to modify it. That is to say, the source code openness doesn’t guarantee that we can access, as artists or users, a level of technological development without a greater programming knowledge. Therefore, given the suggestion that open source programs allow access to musical and artistic creation, this question arises: How open is the open?

4. LAYERS AND LEVELS: ARTISTIC PRACTICE AND TECHNOLOGICAL DEVELOPMENT IN HIGH AND LOW LEVEL

In computer music, artistic practice and technological development are related by writing source code or programming. What distinguishes them is perhaps the way they organise the code and the level at which they operate. Artistic practice tends to an exploratory writing in high level, while technological development to a structured writing at both high and low levels.
To show the relationship between these elements in the space that conforms my research, I have designed a graphic that contains two layers: one represents the artistic practice and the other the technological development. These layers are overlapped and are situated beside the high and low programming levels. The layers and the levels are contained inside an area, marked by a dotted red line, that represents the ambit of source code human expression. The low part of the graphic is framed by a dotted grey line that represents the ambit of the computer and its language. The ambit of the computer executes the ideas expressed by humans in the source code through the translation that interpretation or compilation programs carry out. This article is focused on a problem that occurs in the section framed by the source code –i.e. the dotted red line.

Figure 2: Conceptual space of the research.

The artistic practice layer refers, in my research, to musical practices with creative processes based on written source code such as live coding or algorithmic composition. On the other hand, the technological development layer refers to software creation, programs, objects and libraries to carry out music creation. In this context, both layers are not separated but intertwined, the source code written in the artistic layer is usually of an exploratory nature following the syntactic rules of programming language to combine different program functions but not necessarily constrained to certain structures. The technological layer source code is usually more structured and standardized than the previous one, as it follows programming conventions to make a software operational. In both cases there is a different degree of formalization of the source code writing that requires different knowledge.

Artistic practice expresses ideas and concepts with source code written in a programming language as well as technological development. Both layers – artistic practice and technological development– are related by the same material: the source code. This distinction separates artistic creation from technological development, however, they happen in the same stage and rise the following question:

What would happen if musical creation is developed at the technological layer and at lower programming levels? Perhaps addressing technological layers and lower programming levels from art can detonate ideas and ways to solve artistic problems (and vice versa, to solve technological problems from the artistic layer).

I have tried to incorporate the categories of the graph in my music practice to answer, expand and clarify the research questions that arise from this process. The first step has been to classify the SuperCollider software, which I use in my artistic practice, into levels and layers. As a second step I have developed an object for this software called SonoTexto, which allowed me to explore both layers in the context of improvisation and live coding. The processes mentioned are described below.

5. LEVELS AND LAYERS IN SUPERCOLLIDER

The case I choose to test the categories of levels and layers is the SuperCollider software. The openness of its source code allows me to access its structure, as well as make visible the technological development and the processes of musical creation.

This software involves two programming languages: at the highest level is sclang; with this language are written the code for musical creation and part of the technological development of the program as classes, objects and libraries. At the lowest level is C++, the language in which the program is written, as well as the unit generators and plug-ins. I propose the distribution of the layers as follows: the artistic practice occurs in the highest level, or sclang, while the technological development layer occurs at both levels, sclang and C++.

Baalman (2015) mentions that in a first encounter with a programming language we look for ways to express ourselves, we develop a subset of the language for our daily conversation. The author says that if we are not satisfied with the language we will look for another one or the way to write extensions that allow us to express ourselves to our taste. This idea responds in part to the intention to explore the technological development from the artistic practice. At this point –the passage between layers and levels– I find a problem linked to the relationship between artistic practice and technological development in the field of computer music.
The openness of the SuperCollider source code lets me see how the classes, objects and plug-ins, that constitute it, are written; this allows me to write my own technological objects that extend the functionality of the program. However, to understand the source code of the program at its lowest level, a greater level of programming and development knowledge is required, but: Why move on to a technological layer or lower levels of programming from artistic practice?

The intention to descend levels and pass between layers to access the technological structures of the program allows me to better understand the technology I use in my practice of musical creation and design some functions that allow me to express myself. I can explore, snoop around, try to understand the configuration and the reason for its functions, even if the structure of the program is too complex to understand it in its totality. First of all, I depart from the principle, and the invitation, of an open source code to provoke aesthetic situations in a field supposedly delegated to technological development. Also, it helps me to understand how the aesthetics of computer music is created, which I see closely linked to a programming discourse. That is to say, many times the code or the algorithm itself becomes the artwork, so, is it possible to state that the technological discourse, in computer artistic practices, is the artistic discourse?

A way to explore this problem in my research is the development of an object for SuperCollider called SonoTexto, with which I approach the practice of improvisation from the recording of the sound environment and its manipulation through source code. This approach results in two parts: first, the development of a technological object and second, the use of this object to perform an artistic practice.

6. SONOTEXTO: MUSICAL CREATION BETWEEN LAYERS AND LEVELS

SonoTexto is an object of SuperCollider written with sclang that allows to record, reproduce and modify short fragments of sound. The sound is recorded with a microphone connected to the computer, or the built-in microphone, and stored in a buffer of the temporary memory of the computer, the size of which is defined in advance. Each buffer can be read at different points, change its speed and direction of reading. The reproduction and manipulation of the captured sound is done through written source code that is modified while it runs, a practice known as live coding.

The idea behind SonoTexto is to improvise with the sound emitted by the instrument of another improviser, some sound object or the sound environment of a specific space. These sound materials are recorded at the moment and they sound for the first time in the sound system and in the space where the improvisation occurs. The possibility of recording several sound buffers at the moment allows to incorporate them into the sound environment or the ensemble from which they were captured. The result is a series of superposed sounds that add to the way the acoustic space sounds or to the sound that an improvisation ensemble produces.

In this case, the intervention on sound is not direct, it is mediated by the writing of source code. To record sound I write the word rec and to reproduce it play, in the same way there are a series of commands that indicate different actions and modifications. When improvising with SonoTexto, the artistic and technological layers operate at the same time, in an intertwined way and with the source code as common material, which in turn is translated by the computer to convert the described actions into sound. At the moment of improvising, I act at the high level of the program that allows me to flexibly write the instructions to modify and organise the sound over time. Below the improvisation, there is a technological layer in which SonoTexto operates alongside others objects, unit generators, structures and software.
The research-creation process described allows me to build an object on the technological layer that, later, allows me to create music in the artistic layer. Both layers create a series of relationships in the process of writing code and translate my ideas into sound. The technological object is activated at the time of the improvisation to record sound, this way I can act at a high level of abstraction. In this case, artistic thinking extends to the technological layer at a lower level to reach the middle structures of the program; in this sense, it is possible to activate a technological piece to perform one of musical creation.

The class SonoTexto can be downloaded from https://github.com/hvillase/sonotexto. There are some video examples of SonoTexto: the first one shows an improvisation ensemble performing the rule-base piece Juegos de Improvisación by Nefi Domínguez on 3th May 2019 at Music Faculty UNAM https://youtu.be/0mw3CmoFapw, the second one is an apartment improvisation in Mexico City https://youtu.be/VZpgOv1-naE and the third one is a live coding concert in Galería el Rule in Mexico City on 26th June 2019 https://youtu.be/nt3b20MfC70.

7. CONCLUSIONS

Programming languages and source code have a central role in this research as agents that make visible the relationship between technological development and artistic practice. It seems to me that technological discourse and programming knowledge largely determine computer art and its practices, which is expressed in the artistic language of computer music. The question about the relationship between these categories is not new, it arises perhaps from the moment in which it has been possible to make art with computers. Therefore, I think it is necessary to update these questions and continue to problematise this relationship as artistic practices are transformed and new technologies emerge or they renew.

This text, alongside the presented technological and artistic work, doesn’t fully explore the spectrum between high and low programming levels, however, it opens the possibility to reflect on the technological structures that occur under the artistic practice and the implications that arise from this relationship. The openness of the source code of a program is an invitation to explore it, although it doesn’t guarantee direct access because specialized knowledge is required when we go down levels or when we pass between artistic and technological layers. This research will continue to explore the presented problem from a research approach that is informed by practice and the feedback between artistic practice and technological development.

8. ACKNOWLEDGEMENTS

This text was written and presented with the support of CONACyT and PAEP through the Graduate Music Program of National Autonomous University of Mexico (UNAM).

9. REFERENCES

Baalman, M. (2015) Embodiment of code. In: McLean, A., Magnusson, T., Ng, K., Knotts, S. and Armitage, J. International Conference on Live Coding 2015. Leeds, UK, 13-15 July. Leeds: ICSRiM. 35-40.

Barragán, H. (2007) Software: ¿Arte?. In: La Ferla, J. (ed). El medio es el diseño audiovisual. Manizales: Universidad de Caldas. 583-587.

Berry, D. (2011) The philosophy of software: Code and Mediation in the Digital Age. Hampshire: Palgrave Macmillan.

Bolanakis, D. E., Evangelakis, G.A., Giavas, E. and Kotsis, K. T. (2011) A Teaching Approach for Bridging the Gap Between Low-Level and High-Level Programming Using Assembly Language Learning for Small Microcontrollers. Computer Application in Engineering Education, 19 (3). 525-537.

Chisnall, D. (2018) C Is Not a Low-Level Language: Your computer is not a fast PDP-11. Communications of the ACM, 7 (61). 44-48.

Lee, S.W. (2019) Show Them My Screen: Mirroring a Laptop Screen as an Expressive and Communicative Means in Computer Music. In: Queiroz, M., Sedó, A.X. Proceedings of the International Conference on New Interfaces for Musical Expression. Porto Alegre, Brazil, 3-6 June 2019. UFRGS. 443–448.

Temkin, D. (2017) Language Without Code: Intentionally Unusable, Uncomputable, or Conceptual Programming Languages. Journal of Science and Technology of the Arts, 9 (3). 83-91.

Vazquez-Vilar, M., Orzaez, D. and Patron, N. (2018) DNA assembly standards: Setting the low-level programming code for plant biotechnology. Plant Science, 273. 33-41.