Defining Programming languages for Music through the view of “Somewhat Weak” Computer Music

Anonymized for review

example@example.com

ABSTRACT

In this paper, the author introduces the perspective of “Somewhat Weak Computer Music” in order to describe the history of programming languages for music without being bound by the style of computer music, and conduct a critical review of the history programming languages for music. This paper focuses on a critical review of the post-acousmatic discourse, which is an inclusive notion for recent tendencies in computer music. The universalism associated with pulse-code modulation, which is the basis of sound programming today, has functioned as a discourse that invites expectations of musicians and scientists, even though in reality the range of expression is limited to that era. In addition, the MUSIC-N family, which is the origin of sound generation with a computer based on PCM, is contextualized more as a series of workflows for generating sound on a computer rather than as a semantics and specification of programming languages, and it has gradually developed as a black box that users do not need to understand its internal structure. The author concludes that programming languages for music developed since the 1990s are not necessarily aimed at creating new musical styles, but also have the aspect of presenting an alternative to the technological infrastructure around music, such as formats and protocols which is becoming more invisible, and a new point of discussion is presented for future historical research on music using computers.

1. INTRODUCTION

Programming languages and environments for music have developed hand in hand with the history of creating music using computers. Software like Max, Pure Data, CSound, and SuperCollider has been referred to as “Computer Music Language” [1, 2, 3], “Language for Computer Music” [4], and “Computer Music Programming Systems” [5], though there is no clear consensus on the use of these terms. However, as the term “Computer Music” suggests, these programming languages are deeply intertwined with the history of technology-driven music, which developed under the premise that “almost any sound can be produced” [6] through the use of computers.

In the early days, when computers were confined to university research laboratories and neither displays nor mice existed, creating sound or music with computers was inevitably linked to programming. Today, however, using programming as a means to produce sound on a computer—rather than employing DAW (Digital Audio Workstation) software—is somewhat specialized. In other words, programming languages for music developed after the proliferation of personal computers are software that deliberately choose programming (whether textual or graphical) as their frontend for sound generation.

Since the 1990s, theoretical advancements in programming languages and the various constraints required for real-time audio processing have significantly increased the specialized knowledge needed to develop programming languages for music. Furthermore, some music-related languages developed after the 2000s are not necessarily aimed at pursuing new forms of musical expression. There appears to be no unified perspective on how to evaluate such languages.

The ultimate goal of this paper is to introduce the framework of “weak computer music,” referring to music mediated by computers in a non-style-specific manner. This framework aims to decouple the evaluation of programming language design and development for music from specific styles and the ideologies associated with computer music.

1.1 Use of the Term “Computer Music”

Despite its potential broad application, the term “computer music” has been repeatedly noted since the 1990s as being used within a narrowly defined framework, tied to specific styles or communities [7].

The necessity of using the term “computer music” for such academic contexts (particularly those centered around the International Computer Music Conference, or ICMC) has diminished over time. Lyon argues that defining computer music as simply “music made using computers” is too permissive, while defining it as “music that could not exist without computers” is overly strict, complicating the evaluation of analog modeling synthesizers implemented on computers. Lyon questions the utility of the term itself, comparing its consideration to that of “piano music,” which ignores the styles within it [8].

As Ostertag and Lyon observed, it has become increasingly difficult to envision a situation where computers are absent from the production and experience of music today, particularly in commercial contexts [1]. Nevertheless,

[1] Of course, the realm of music extends beyond the numbers processed

the majority of music in the world could be described as "simply using computers."

Holbrook and Rudi propose analyzing what has been called computer music within the framework of post-acousmatic music [9], including traditions of pre-computer electronic music as one of many forms of technology-based/driven music [10].

A critical issue with these discussions is that post-acousmatic music lacks a precise definition. One proposed characteristic is the shift in the locus of production from institutions to individuals, which has altered how technology is used [9, p113]. However, this narrative incorporates a tautological issue: while it acknowledges the historical fact that the decreasing cost of computers allowed diverse musical expressions outside laboratories, it excludes much music as "simply using computers" and fails to provide insights into such divisions.

The spread of personal computers has incompletely achieved the vision of metamedium as a device users could modify themselves, instead becoming a black box for content consumption [11]. Histories highlighting the agency of those who created programming environments, software, protocols, and formats for music obscure indirect power relationships generated by the infrastructure [12].

Today, while music production fundamentally depends on computers, most of it falls under Lyon's overlapping permissive and strict definitions of computer music. In this paper, I propose calling this situation the following:

> "Weak computer music" — music for which computers are essential to its realization, but where the uniqueness of the work as intended by the creator is not particularly tied to the use of computers.

Most people use computers simply because no quicker alternative exists, not because they are deliberately leveraging the unique medium of computers for music production. However, the possibility that such music culture, shaped by the incidental use of computers, has aesthetic and social characteristics worth analyzing cannot be dismissed.

This paper will historically organize the specifications and construction of programming languages for early computer music systems with a focus on their style-agnostic nature.

- Examining the discourse framing MUSIC as the progenitor of computer music.
- Investigating what aspects were excluded from user access in MUSIC-N derivatives such as MUSIGOL.
- Analyzing the standardization of UGens (unit generators) and the division of labor in Max and Pure Data.
- Reviewing music programming languages of the 2000s.

The conclusion will propose a framework necessary for future discussions on music programming languages.

by computers or the oscillations of speaker diaphragms. This paper does not seek to intervene in aesthetic judgments regarding music made without computers or non-commercial musical activities. However, the existence of such music does not counter the awareness that there is little analysis of the inevitable involvement of computing as a medium in the field of popular music, which attracts significant academic and societal interest.

2. BORN OF "COMPUTER MUSIC" - MUSIC-N AND PCM UNIVERSALITY

Among the earliest examples of computer music research, the MUSIC I system (1957) from Bell Labs and its derivatives, known as MUSIC-N, are frequently highlighted. However, attempts to create music with computers in the UK and Australia prior to MUSIC I have also been documented [13].

Organizing what was achieved by MUSIC-N and earlier efforts can help clarify definitions of computer music.

The earliest experiments with sound generation on computers in the 1950s involved controlling the intervals between one-bit pulses (on or off) to control pitch. This was partly because the operational clock frequencies of early computers fell within the audible range, making the sonification of electrical signals a practical and cost-effective debugging method compared to visualizing them on displays or oscilloscopes. Computers like Australia's CSIR Mark I even featured primitive instructions like a "hoot" command to emit a single pulse to a speaker.

In the UK, Louis Wilson discovered that an AM radio near the BINAC computer picked up electromagnetic waves generated by vacuum tube switching, producing regular tones. This serendipitous discovery led to the intentional programming of pulse intervals to generate melodies [14].

However, not all sound generation prior to PCM (Pulse Code Modulation) was merely the reproduction of existing music. Doornbusch highlights experiments on the British Pilot ACE (Prototype for Automatic Computing Engine: ACE), which utilized acoustic delay line memory to produce unique sounds [13, p303-304]. Acoustic delay line memory, used as main memory in early computers like BINAC and CSIR Mark I, employed the feedback of pulses traveling through mercury via a speaker and microphone setup to retain data. Donald Davis, an engineer on the ACE project, described the sounds it produced as follows [15, p19-20]:

> The Ace Pilot Model and its successor, the Ace proper, were both capable of composing their own music and playing it on a little speaker built into the control desk. I say composing because no human had any intentional part in choosing the notes. The music was very interesting, though atonal, and began by playing rising arpeggios: these gradually became more complex and faster, like a developing fugue. They dissolved into colored noise as the complexity went beyond human understanding.
>
> Loops were always multiples of 32 microseconds long, so notes had frequencies which were submultiples of 31.25 KHz. The music was based on a very strange scale, which was nothing like equal tempered or harmonic, but was quite pleasant. This music arose unintentionally during program optimization and was made possible by "misusing" switches installed for debugging acoustic delay line memory (p20).

Media scholar Miyazaki described the practice of listening to sounds generated by algorithms and their bit patterns, integrated into programming and debugging, as “Algo*rhythmic* Listening” [16].

Doornbusch warns against ignoring early computer music practices in Australia and the UK simply because they did not directly influence subsequent research [13, p305]. Indeed, the tendency to treat pre-MUSIC attempts as hobbyist efforts by engineers and post-MUSIC endeavors as serious research remains common even today [17].

The sounds generated by Pilot ACE challenge the post-acousmatic narrative that computer music transitioned from laboratory-based professional practices to personal use by amateurs. This is because: 1. The sounds were produced not by music specialists but by engineers, and 2. The sounds were tied to hardware-specific characteristics of acoustic delay line memory, making them difficult to replicate even with modern audio programming environments. Similarly, at MIT in the 1960s, Peter Samson utilized a debug speaker attached to the aging TX-0 computer to experiment with generating melodies using square waves [18].

This effort evolved into a program that allowed users to describe melodies with text strings. For instance, writing `4fs t8` would produce an F4 note as an eighth note. Samson later adapted this work to the PDP-1 computer, creating the “Harmony Compiler,” widely used by MIT students. He also developed the Samson Box in the early 1970s, a computer music system used at Stanford University’s CCRMA for over a decade [19]. These examples suggest that the initial purpose of debugging does not warrant segregating early computational sound generation from the broader history of computer music.

2.1 Universality of PCM

Let us examine **Pulse Code Modulation (PCM)**—a foundational aspect of MUSIC’s legacy and one of the key reasons it is considered a milestone in the history of computer music. PCM enables the theoretical representation of “almost any sound” on a computer by dividing audio waveforms into discrete intervals (sampling) and expressing the amplitude of each interval as quantized numerical values. It remains the fundamental representation of sound on modern computers. The underlying sampling theorem was introduced by Nyquist in 1928 [20], and PCM itself was developed by Reeves in 1938.

A critical issue with the “post-acousmatic” framework in computer music history lies within the term “acousmatic” itself. Initially proposed by Piegnot and later theorized by Schaeffer, the term describes a mode of listening to tape music, such as musique concrète, in which the listener does not imagine a specific sound source. It has been widely applied in theories of recorded sound, including Chion’s analyses of sound design in visual media.

However, as sound studies scholar Jonathan Sterne has pointed out, discourses surrounding acousmatic listening often work to delineate pre-recording auditory experiences as “natural” by contrast [2]. This implies that prior to the advent of recording technologies, listening was unmediated and holistic—a narrative that obscures the constructed nature of these assumptions.

> For instance, the claim that sound reproduction has “alienated” the voice from the human body implies that the voice and the body existed in some prior holistic, unalienated, and self present relation.
>
> They assume that, at some time prior to the invention of sound reproduction technologies, the body was whole, undamaged, and phenomenologically coherent. [22, p20-21]

The claim that PCM-based sound synthesis can produce “almost any sound” is underpinned by an ideology associated with recording technologies. This ideology assumes that recorded sound contains an “original” source and that listeners can distinguish distortions or noise from it. Sampling theory builds on this premise by statistically modeling human auditory characteristics: it assumes that humans cannot discern volume differences below certain thresholds or perceive vibrations outside specific frequency ranges. By limiting representation to this range, sampling theory ensures that all audible sounds can be effectively encoded.

By the way, the actual implementation of PCM in MUSIC I only allowed for monophonic triangle waves with controllable volume, pitch, and timing (MUSIC II later expanded this to four oscillators) [23]. Would anyone today describe such a system as capable of producing “infinite variations” in sound synthesis?

Even when considering more contemporary applications, processes like ring modulation (RM), amplitude modulation (AM), or distortion often generate aliasing artifacts unless proper oversampling is applied. These artifacts occur because PCM, while universally suitable for reproducing recorded sound, is not inherently versatile as a medium for generating new sounds. As Puckette has argued, alternative representations, such as collections of linear segments or physical modeling synthesis, present other possibilities [24]. Therefore, PCM is not a completely universal tool for creating sound.

Acknowledgments

At the end of the Conclusions, acknowledgments to people, projects, funding agencies, etc. can be included after the second-level heading “Acknowledgments” (with no numbering).

3. REFERENCES

[1] J. McCartney, “Rethinking the Computer Music Language: SuperCollider,” *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, Dec. 2002.

[2] H. Nishino and R. Nakatsu, “Computer Music Languages and Systems: The Synergy Between Technology and Creativity,” in *Handbook of Digital Games and Entertainment Technologies*, 2016.

[3] A. McPherson and K. Tahlroğlu, “Idiomatic Patterns and Aesthetic Influence in Computer Music Languages,” *Organised Sound*, vol. 25, no. 1, pp. 53–63, 2020.

[2] Sterne later critiques the phenomenological basis of acousmatic listening, which presupposes an idealized, intact body as the listening subject. He proposes a methodology of political phenomenology centered on impairment, challenging these normative assumptions [21]. Discussions of universality in computer music should also address ableism, as seen in the relationship between recording technologies and auditory disabilities.

[4] R. B. Dannenberg, “Languages for Computer Music,” *Frontiers in Digital Humanities*, vol. 5, Nov. 2018.

[5] V. Lazzarini, “The Development of Computer Music Programming Systems,” *Journal of New Music Research*, vol. 42, no. 1, pp. 97–110, 2013.

[6] M. V. Mathews, “An Acoustic Compiler for Music and Psychological Stimuli,” *The Bell System Technical Journal*, vol. 40, no. 3, pp. 677–694, May 1961.

[7] B. Ostertag, “Why Computer Music Sucks,” https://web.archive.org/web/20160312125123/http://bobostertag.com/writings-articles-computer-music-sucks.htm, 1998.

[8] E. Lyon, “Do we still need computer Music?” https://disis.music.vt.edu/eric/LyonPapers/Do_We_Still_Need_Computer_Music.pdf, 2006.

[9] M. Adkins, R. Scott, and P. A. Tremblay, “Post-Acousmatic Practice: Re-evaluating Schaeffer’s Heritage,” *Organised Sound*, vol. 21, no. 2, pp. 106–116, Aug. 2016.

[10] U. Holbrook and J. Rudi, “Computer Music and Post-Acousmatic Practices: International Computer Music Conference 2022,” in *Proceedings of the International Computer Music Conference, ICMC 2022*, ser. International Computer Music Conference, ICMC Proceedings, G. Torre, Ed. San Francisco: International Computer Music Association, Jul. 2022, pp. 140–144.

[11] L. Emerson, *Reading Writing Interfaces: From the Digital to the Bookbound*. Univ of Minnesota Press, Nov. 2014.

[12] J. Sterne, “There Is No Music Industry,” *Media Industries Journal*, vol. 1, no. 1, pp. 50–55, Jan. 2014.

[13] P. Doornbusch, “Early Computer Music Experiments in Australia and England,” *Organised Sound*, vol. 22, no. 2, pp. 297–307, Aug. 2017.

[14] R. D. Woltman, F. B. Woltman, L. D. Wilson, A. B. Tonik, J. K. Swearingen, C. M. Shuler, J. E. Sberro, J. E. Sammet, H. W. Matter, D. W. Marquardt, F. K. Koons, M. W. Huff, F. E. Holberton, C. Hammer, D. B. Dixon, E. L. Delves, G. Danehower, M. P. Chinitz, L. S. Carter, J. Bartik, L. W. Armstrong, D. P. Armstrong, and A. E. Adams, “UNIVAC Conference.” Charles Babbage Institute, Tech. Rep., 1990.

[15] D. Davis, “Very Early Computer Music,” *Resurrection The Bulletin of the Computer Conservation Society*, vol. 10, pp. 19–20, 1994.

[16] S. Miyazaki, “Algorhythmic Listening 1949-1962 Auditory Practices of Early Mainframe Computing,” in *AISB/IACAP World Congress 2012: Symposium on the History and Philosophy of Programming, Part of Alan Turing Year 2012*, 2012, p. 5.

[17] H. Tanaka, *All About Chiptune: New Music Born from Games*. Seibundo Shinkosha, 2017.

[18] S. Levy, *Hackers: Heroes of the Computer Revolution - 25th Anniversary Edition*, 1st ed. O’Reilly Media, May 2010.

[19] D. G. Loy, “Life and Times of the Samson Box,” *Computer Music Journal*, vol. 37, no. 3, pp. 26–48, 2013.

[20] H. Nyquist, “Certain Topics in Telegraph Transmission Theory,” *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, Apr. 1928.

[21] J. Sterne, *Diminished Faculties: A Political Phenomenology of Impairment*. Durham: Duke Univ Press, Jan. 2022.

[22] ——, *The Audible Past: Cultural Origins of Sound Reproduction*. Durham: Duke University Press, 2003.

[23] M. Mathews and C. Roads, “Interview with Max Mathews,” *Computer Music Journal*, vol. 4, no. 4, pp. 15–22, 1980.

[24] M. Puckette, “The Sampling Theorem and Its Discontents,” *International Computer Music Conference*, pp. 1–14, 2015.