CRICKET TOOLS: REAL-TIME AUDIO SCORING OF QUANTUM STATE EVOLUTION ACROSS NETWORKED MOBILE DEVICES

Pedro González-Fernández
Escola Superior de Música de Catalunya, Barcelona
pgonzalez2@esmuc.cat

Thomas Noll
Escola Superior de Música de Catalunya, Barcelona
tnoll@esmuc.cat

Reiko Yamada
Escola Superior de Música de Catalunya / ICFO - Institute of Photonic Science, Barcelona
ryamada@esmuc.cat

ABSTRACT

This paper presents Cricket Tools, a browser based client-server system for distributed, real-time audio scoring and synchronized audio playback across mobile devices (phones, tablets) connected to the same local network. Cricket Tools supports two complementary approaches: (1) synchronized fixed-media playback (e.g., multi-part stems) and (2) parameterized, real-time audio scores in which each performer's device synthesizes an individualized audio guide. This second paradigm is designed for situations where the "score" is not a pre-rendered audio or video file but an evolving process controlled by a conductor or an interactive generative system.

In the paper, we position Cricket Tools relative to prior work on audio and audiovisual scoring and distributed notation, drawing on Bhagwati's framework of audio scores and on Schimana's "sound as score" practice, as well as on Bell's SmartVox/SmartBox lineage of web-based distributed parts for ensemble rehearsal and performance.

We also describe the system's architecture and timing approach, including a lightweight time-synchronization model and client-side scheduling for consistent playback across devices. As a use case, we outline a prototype in which features derived from quantum wavefunction evolution (magnitude/phase descriptors) are mapped to real-time audiovisual scoring parameters distributed to multiple players.

1. INTRODUCTION

1.1 Context and Motivation

Digital notation and time-based score media have broadened the idea of what a "score" can be, especially in screen-based cases where time, motion and interaction become part of the notation itself. In animated notation, we follow Fischer´s understanding of the score as an invitation to a *mapping process* in which performers connect visual attributes to sonic or performative parameters, often using a screen-based temporal reference such as a moving playhead [1].

Within this broader shift, the sound itself has also been treated as a notational medium. Schimana explicitly frames "sound as score" as a way to communicate with musicians through listening rather than written notation, including approaches such as "live generated audio score" (performers imitate sounds they hear) and acoustic memory-based scoring methods [2].

In a related way, Bhagwati proposes "audio scores" as an interface for musical conveyance through sound and develops a vocabulary for how such scores operate in practice and rehearsal. He proposes a set of conveyance modes: information, instruction, imitation, inspiration, and instance; and discusses how indexical instructions can frame other modes. Alongside these classification, Bhagwati notes that audio scores can be implemented as live-generated material or as fixed, repeatable individual tracks, and he identifies synchronization as a remaining technical uncertainty in more "data network-centric" situations with many networked devices [3].

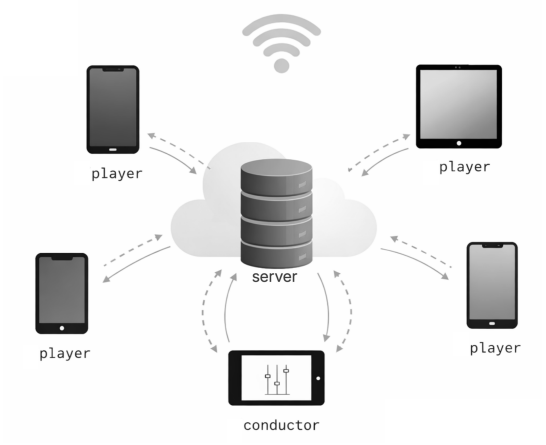

Figure 1. Local-network setup with conductor and multiple player devices connected to a central server.

1.2 Distributed and Networked Scoring on Mobile Devices

A central motivation that follows from these approaches to notation is practical: once a score becomes time-based (audio, audiovisual, animated, or otherwise dynamic), performers need a shared temporal reference while still being

able to work with individualized parts. In work on networked digital scores, Eldridge et al. highlight synchronization between conductor, performers, and notation as a core design issue, alongside performer autonomy and usability in rehearsal [4].

Bell's SmartVox/SmartBox lineage demonstrates a web-based approach to distributing synchronized parts to phones and tablets using separate performer and conductor clients, where the conductor can start/stop playback and jump to labeled sections; Bell also documents earlier approaches such as iVideoShow (a native iOS app remotely controlled via OSC over Wi-Fi) [5] [6].

More broadly, a substantial body of related work has developed distributed systems for screen-centered score delivery and dynamic visual rendering over local networks, including InScore[7], Quintet.net [8], NetCanvas[9], and ZScore[10], as well as MaxScore[11] and Bach-based[12] workflows, Decibel ScorePlayer[13], dfscore[14], and integrated environments such as Antescofo[15]; middleware-oriented frameworks such as Odot[16] (including OpenMusic integration) or Landini. Our system is situated within this same distributed scoring problem space, but is designed as an audio-score-centered tool.

Figure 2. Cricket Tools landing page.

1.3 Cricket Tools

Cricket Tools is a browser-based client–server system for local-network performances where several participants (performers, or audience members) use their own mobile devices as sound score receivers. Its design follows two established needs already present in the literature: (1) reliable synchronized playback of distinct fixed-media parts (as in SmartVox's distributed parts and earlier mobile playback approaches iVideoShow), and (2) support for "live-generated" or process-driven audio scores where the notational content is produced during performance rather than pre-rendered.

The present paper focuses on the design and technical implementation of Cricket Tools and presents, as an illustrative use case, a prototype mapping of quantum wavefunction evolution descriptors (e.g., magnitude/phase-derived features) into real-time scoring parameters distributed to multiple players, reflecting the original motivation of the project as a distributed quantum sonification tool [17]. The concluding section discusses prospective extensions of the system toward more complex parameterized scores, including microtonal notation and haptics.

2. IMPLEMENTATION

2.1 Overview

The system is organized around two user roles: Conductor and Player, and two tool modes. The first mode (Synth Tool) distributes a parameterized, real-time audio score in which each Player device synthesizes an individualized track accompanied by a simple animated visualization. The second mode (MP3 Sync) distributes pre-rendered fixed media parts to Players and triggers synchronized playback across devices. Both modes share the same networking layer (Socket.IO).

2.2 Software Architecture

Cricket Tools implements a client–server architecture with a Node.js backend and a browser-based frontend. On the server side, Express provides HTTP routing, while Socket.IO provides persistent bidirectional WebSocket communication for real-time state synchronization. The client is a React single-page application bundled with Vite, served by the same Express instance. All conductor dashboards and player interfaces connect to the server through Socket.IO and receive state updates as events rather than through polling. Fixed-media operations (upload/download/delete) are implemented as HTTP endpoints, while real-time performance controls and state changes propagate through Socket.IO events. Server state is held in memory (players, MP3 slots, and current playback state), which simplifies lookup and broadcast but implies that state resets on server restart.

The implementation uses a simple clock synchronization mechanism to estimate each client's offset from server time. The client sends a syncPing containing a local timestamp; the server replies with syncPong including the original client time and its own server timestamp. The client measures round-trip time and estimates the server time at reception assuming approximately symmetric network delays, then stores the resulting serverTimeOffset. After this step, clients estimate the "current server time" as Date.now() + serverTimeOffset, and use this estimate when scheduling audio events. This offset is used in both tools: in the Synth Tool it supports phase-aligned cyclic triggering; in MP3 Sync it enables scheduling playback against a future server start time with a safety buffer.

2.3 Synth Tool: Distributed Real-Time Audio Scoring

In the Synth Tool, Players join through a browser interface by entering a display name; the server assigns each Player a stable numeric identifier (1, 2, 3, . . .) for the duration of the session. This provides a consistent address-

ing scheme for both the conductor dashboard and external control (Section 2.4). Each Player's state includes at minimum: playerId, socketId, name, pitch (MIDI), interval (cycle duration in ms), and a phaseStartServerTime used for phase alignment.

The conductor dashboard displays all connected Players and provides real-time controls for at least two parameters per Player: pitch (e.g., MIDI 36–84) and interval (e.g., 50–3000 ms). When the conductor changes a control, the client emits a Socket.IO message to the server. The server updates the in-memory Player state and broadcasts (i) a targeted update to the affected Player and (ii) a global state update to all conductor clients. This design ensures that the conductor interfaces remain consistent.

On the Player side, audio is generated locally using Tone.js. Each Player instantiates a synthesizer and produces discrete note events based on the current phase of a cyclic timeline. The phase is computed from estimated server time and the server-provided phaseStartServerTime and interval, so that Players sharing the same phase reference can align their trigger moments even if they joined at different times. In the current prototype, Players also render a circular visual representation of the cycle, supporting the interpretation of the audio guide.

A key implementation detail concerns continuity when the conductor changes tempo/interval. Naively replacing the interval would shift phase and produce discontinuities in timing. To preserve continuity, the server computes the Player's current phase under the old interval and adjusts phaseStartServerTime so that the Player continues from the same phase position under the new interval. This allows performers to experience parameter changes as continuous transformations rather than abrupt resets, while keeping the phase model consistent across conductor and Player clients.

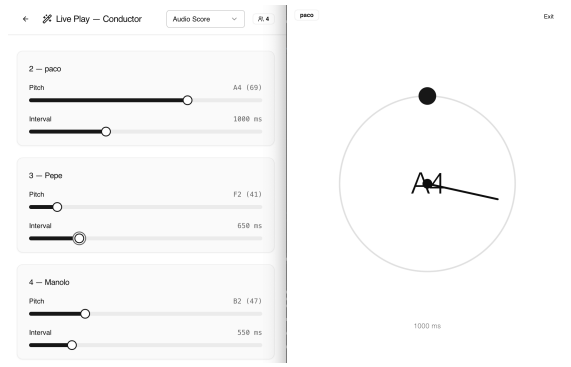

Figure 3. Synth Tool interface showing conductor controls and player view for real-time audio scoring.

2.4 Control from Max/MSP: Node-For-Max Bridge

A core design goal is to allow algorithmic control from composition environments without requiring manual interaction with the conductor dashboard. Cricket Tools provides a Node for Max module (conductor-control.js) that connects to the server via Socket.IO and registers itself as a conductor client on connection. Once connected, the script accepts Max messages and translates them into Socket.IO control events (maxCommand) that follow a compact integer format: target, control, value. The target selects a specific Player by numeric playerId or broadcasts to all Players (e.g., target = -1). The control selects a parameter (e.g., pitch or interval), and value provides the new parameter value.

On the server, incoming maxCommand events are validated as conductor-originated, then routed through a control resolver that maps control IDs to named parameters and applies the update to the addressed Player(s). The server maintains a bidirectional mapping between numeric player IDs and socket IDs, so that a Max patch can address Players consistently even as their socket connections change. The bridge also supports bidirectional feedback by receiving server state updates and outputting basic state information to Max (e.g., the number of connected Players), enabling patches to adapt to the number of players in the ensemble. An OSC interface can also be supported on the server side via UDP, but in the current implementation the Node-for-Max WebSocket bridge is the primary external control pathway.

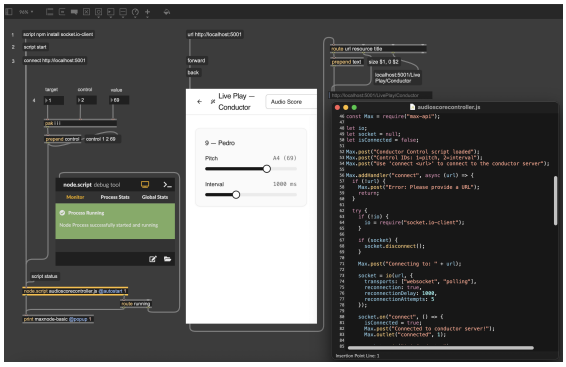

Figure 4. Node-for-Max WebSocket interface enabling algorithmic control of the Synth Tool from Max/MSP.

2.5 Mp3 Sync Tool

The MP3 Sync Tool distributes per-part MP3 files and triggers synchronized playback. It uses a fixed slot model (eight slots), where each slot can hold one player and one assigned file; file assignment persists across player disconnects. The conductor uploads files via HTTP and Players download and decode them locally. For playback, the server broadcasts a future start time and optional seek position; Players schedule Web Audio playback using their estimated server-time offset. Late joiners can be instructed to start from the current playhead derived from elapsed time since the server start.

3. USE CASE: "COLLECTIVE PERFORMANCE OF QUANTUM WAVE FUNCTIONS"

In the present section we outline a specific and inventive scenario of an explorative musical performance, which seeks a particular interaction between a social and a mathematical concept of unity. In classical chamber music (like a

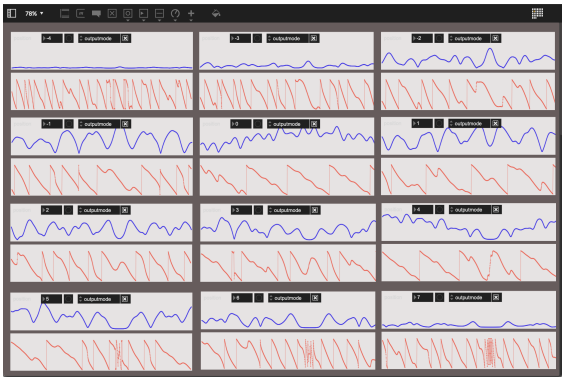

Figure 5. Max/MSP patch visualizing quantum wavefunction evolution (magnitude and phase) and mapping it to real-time audio-score control data.

string quartet), the goal is often the homogenization of sound—the four instruments striving to sound like one single, 16-stringed instrument. Our quantum model takes this "social unity" and provides it with a mathematical backbone: the Non-locality of a quantum wave function.

What we call the *use case* here, is actually first and conceptually a reduced experiment, which serves as an intermediate goal for a more advanced project. It is nevertheless worthwhile to sketch the overall idea, not least thanks to prominent logistic role of the Cricket Tools.

3.1 What does it mean to perform a one-dimensional wave function?

Imagine n musicians sitting in a circle. Each musician represents a single point in space (a position x in the set $\mathbb{Z}_n$). The "one-dimensional" aspect simply means we are looking at a line of data that wraps around—like a string that starts at Musician 1 and ends back at Musician n. At each moment the time-development of the wave function provides each musician with a complex number $\Psi(x,t) = ae^{ip}$, which can be represented by a magnitude $a \geq 0$ and a phase $-\pi \leq p < \pi$. Together these values represent a global unity which is expressed by the Hamilton operator of an underlying quantum system.

In "old-fashioned" chamber music, unity is achieved through a collective interpretation of a score, through empathic listening to each other, through gestures and eye contact. From this tradition we inherit a kind "Non-Soloist" Ethos for the present project: Just as in a Haydn quartet where the inner voices (viola/second violin) are vital for the texture even when they do not have the melody, the musicians at positions x where $|\Psi(x,t)|^2$ is low are still essential. They carry the phase, which is the "potential" for the next move. A shared wavefunction acts as a mathematical version of the "breath" that chamber musicians use to start a phrase together and which lets the ensemble act as a single organism. In an earlier contribution ([17] we developed ideas and tools for the musical sonification of quantum wave functions with the means of computer generated sounds. In our Max/MSP package *Qte* one may specify a Hamiltonian H of a (disturbed) Harmonic oscillator, calculate the eigenstates $\phi_n(x)$, choose an initial state ψ and store its time development $\Psi(x,t)$ as a parameter family for the combined control of a family of tracks in a sequencer[18].

With the present approach we meet the following challenge: How can we inherit the spirit of the social interaction in an ensemble performance on the basis of a fully determined time development — where at first sight the musicians would be degraded to be oscillators for the computer-generated flow of data? The answer is obvious: The musicians should not only share the wave function but they should internalize the entire quantum system from the Hamiltonian H, its eigenstates $\phi_n(x)$, the choice of an initial state $\psi(x)$ in order to know and experience the wavefunction $\Psi(x,t)$ as their collective creation. In combination with the *Qte*-package the *Cricket Tools* will help the musicians to immerse themselves as an ensemble into this process of (self)-discovery.

3.2 The Hamiltonian as the Rulebook of Relationships for the Quantum-Ensemble

The "collective rehearsal" starts with the collective design of a Hamilton-operator H(instead of using a predefined operator from the physics classroom). In the position basis, the Hamiltonian H is an $n \times n$ matrix. Its coefficients can be divided into two categories: the Diagonal and the Off-Diagonal. It is important to distinguish between the meanings of the position and the attitudes of the players at these positions. For example, the real diagonal-coefficient H_{xx} at position x represents the "Potential Energy" at that spot. It is the "Gravity" or "Persistence" of this position. If H_{xx} is high compared to others, that position is "expensive" for the wave to stay in. The wave, i.e. the music, will want to move away quickly. If all H_{xx} are the same, the circle is "flat" and perfectly symmetrical. In other words, the diagonal of H is the "topography" of the stage. High tension acts like an obstacle; low tension acts like a welcoming valley for the music to settle in.

The off-diagonal coefficients represent the interaction dispositions of the ensemble. These are complex numbers and high absolute values H_{xy} stand for a higher degree of "openness" between the player at x and the player at y. It determines how fast music is transmitted between the two. The absolute values coincide $|H_{xy}| = |H_{yx}|$ which means that the amount of sending and attentive receiving have to coincide. The actual direction depends on various factors, such as the initial state. But it can also be influenced by the inverse phases of H_{xy} and H_{yx}, through the activation of a phase rotation which induces a direction of the flow on the circle.

For the more advanced project we envisage a time-dependent Hamiltonian, where a small ensemble may change its interaction rules during the performance. For the present use-case we envisage a larger ensemble with a fixed and simplified Hamiltonian[19].

3.3 Our Scenario: The Homogeneous Circle-Ensemble

This scenario investigates the realization of the idea within a large ensemble arranged in a periodic lattice. The goal is to exemplify the movement of a quantum wave packet

through a medium of uniform tension and symmetric coupling. The ensemble of n musicians is mapped onto the cyclic group $\mathbb{Z}_n$. The collective state $|\psi(t)\rangle$ exists in an n-dimensional Hilbert space $\mathcal{H} \cong \mathbb{C}^n$:

$$|\psi(t)\rangle = \begin{pmatrix} c_0(t) \\ c_1(t) \\ \vdots \\ c_{n-1}(t) \end{pmatrix}$$

where each coefficient $c_x(t) = \langle j|\psi(t)\rangle$ provides the specific performance instructions for musician x.

To model a uniform communicative disposition, we employ a nearest-neighbor Hamiltonian. For an ensemble of size n, the matrix H is defined as:

$$H = \begin{pmatrix} E_0 & -ve^{i\phi} & 0 & \dots & -ve^{-i\phi} \\ -ve^{-i\phi} & E_1 & -ve^{i\phi} & \dots & 0 \\ 0 & -ve^{-i\phi} & E_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -ve^{i\phi} & 0 & 0 & \dots & E_{n-1} \end{pmatrix}$$

We interpret the parameters as follows:

- **Diagonal ($E_0, \dots, E_{n-1}$):** Represents the "Tension" of the stage. In the special case $E_0 = \dots = E_{n-1}$ the distribution of potential energy on the stage is homogenous.
- **Off-Diagonal (v):** The coupling strength or "Bridge." It determines the speed at which the music flows between neighbors.
- **Phase (ϕ):** The "Chiral Current." Because $H_{j,j+1} = (H_{j+1,j})^*$, the reciprocity is preserved, but a nonzero ϕ breaks time-reversal symmetry, causing the wave to "spin" directionally around the circle.

3.4 Logistics of the communication with the cricket tools

We assume that each player has her/his mobile phone equipped with a set of headphones, and connects via our QR-code to the local network (via : $// < server - ip >$: 5001) and the cricket-tool-interface in their browser. They access the real-time synth tool, they type their name, receive their player number and accept the usage of the audio system in their phones. In this use case only the conductor can access the Max/MSP-Qte package and the Node-for-Max WebSocket bridge with all the specifications of the Hamiltonian, the initial state and the scorification (parameter settings for the time developments and the actual sound scoring).

4. CONCLUSIONS AND FUTURE WORK

In the quantum-wavefunction use case, Cricket Tools was used in a deliberately reduced setup to distribute magnitude and phase derived descriptors as real-time scoring parameters to multiple players. Within this constrained scenario, the workflow suggested that parameter-stream–based audio scoring can function in a music ensemble setup: performers were generally able to follow the cyclic cueing logic and respond to conductor-side parameter changes with ease. Future work will therefore focus on extending the affordances of the system while preserving interpretability and coherence across devices.

Beyond the current parameter set, Cricket Tools could be extended to support tuning- and intonation-sensitive scoring where pitch is specified not only as a note name but as an explicit mapping to exact frequencies. One concrete direction is to map the same set of twelve note names onto multiple temperament systems, so that the exact frequencies associated with a given pitch class are recalculated whenever the tuning changes. This would turn the distributed audio score into a "tuning-aware" guide in which performers receive not only pitch information but an explicitly defined intonation context, enabling controlled microdeviations that remain coherent across the ensemble.

Musically, such tuning transformations could be deployed as recurring formal events: short, clearly delimited sections in which the distributed parts shift temperament in a coordinated way, alternating with intervening passages based on pre-written material that remains fixed within a single tuning system. To heighten perceptual contrast and stabilize the listening frame, one performer (or a dedicated electronic reference part) could remain in a constant tuning throughout the piece, functioning as an anchor against which each reappearance of the "variable temperament" sections becomes newly legible. This proposal remains exploratory, but it suggests how the same infrastructure used for real-time parameter distribution can also support compositional strategies based on repeatable, structurally articulated changes in intonation.

From an implementation standpoint, this extension can be treated as a direct generalization of the current Synth Tool parameter model. Alongside per-player pitch and interval, the server would maintain a global (or per-player) tuning state, selecting a temperament preset or a transmitted tuning table that maps the twelve pitch classes to cents offsets (relative to 12-TET) or to absolute frequencies. To preserve musical clarity and ensemble stability, tuning changes should be scheduled rather than applied immediately: the server can quantize a retuning event to a musically meaningful boundary (e.g., the next cycle onset) or broadcast it as a future server-time update, reusing the same timing approach already employed for phase alignment and for continuity under interval changes. In this way, retuning is perceived as a controlled transformation rather than an abrupt reset, while remaining consistent across devices.

The Node-for-Max bridge provides a natural pathway for algorithmic intonation control. Instead of manual conductor intervention, a Max/MSP process can compute and send tuning changes as part of the same control stream that already addresses player parameters: selecting between predefined temperament families, interpolating between tuning tables over a specified duration, or generating custom microtonal offsets based on higher-level musical logic. In

the quantum-wavefunction use case outlined in this paper, this creates a particularly direct expansion of the mapping space: magnitude- and phase-derived descriptors could be used not only to drive timbral or rhythmic parameters, but also to shape an ensemble-wide "intonation field" (e.g., phase features selecting temperament regions or specifying continuous cents deviations), while the fixed-tuning reference part provides a perceptual baseline against which such changes can be heard as form-defining shifts.

Finally, because microtonal retuning is both perceptually subtle and cognitively demanding, future versions could strengthen multimodal cueing around tuning events. Player interfaces could display a concise indicator of the current temperament (name and/or deviation profile) and an anticipatory cue when a tuning change is scheduled (e.g., a countdown ring aligned to the cycle). Where devices allow it, brief haptic cues could reinforce the moment of retuning without requiring constant screen attention. Combined with Cricket Tools' dual paradigm (live-generated audio guides in the Synth Tool and fixed reference materials in MP3 Sync) these additions point toward hybrid scores in which intonation functions as a repeatable, structurally articulated parameter: precisely timed, algorithmically controllable, perceptually anchored, and distributed reliably across a networked ensemble.

5. REFERENCES

[1] C. M. Fischer, "Understanding animated notation," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2015*, Paris, France, 2015, pp. 31–38.

[2] E. Schimana, "Sound as score: Live generated audio score and audio score based on acoustic memory," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2019*, Melbourne, Australia, 2019, pp. 33–37.

[3] S. Bhagwati, "Elaborate audio scores: Concepts, affordances and tools," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2018*, Montreal, Canada, 2018, pp. 24–32.

[4] A. Eldridge, E. Hughes, and C. Kiefer, "Designing dynamic networked scores to enhance the experience of ensemble music making," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2016*, Cambridge, UK, 2016, pp. 193–199.

[5] J. Bell and B. Matuszewski, "Smartvox. a web-based distributed media player as notation tool for choral practices," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2017*, A Coruña, Spain, 2017, pp. 99–104.

[6] J. Bell, "Audiovisual scores and parts synchronized over the web," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2018*, Montreal, Canada, 2018, pp. 17–23.

[7] D. Fober, S. Letz, Y. Orlarey, and F. Bevilacqua, "Programming interactive music scores with inscore," in *Proc. Sound and Music Computing Conf. (SMC 2013)*, Stockholm, Sweden, 2013, pp. 185–190.

[8] G. Hajdu, "Quintet.net: An environment for composing and performing music on the internet," *Leonardo Music Journal*, vol. 15, pp. 23–30, 2005.

[9] B. Carey and G. Hajdu, "Netscore: an image server/client package for transmitting notated music to browser and virtual reality interfaces," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2016*, Cambridge, UK, 2016, pp. 151–156.

[10] S. Zagorac and P. Alessandrini, "Zscore: A distributed system for integrated mixed music composition and performance," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2018*, Montreal, Canada, 2018, pp. 62–70.

[11] N. Didkovsky and G. Hajdu, "Maxscore: Music notation in max/msp," in *Proc. Int. Computer Music Conf. (ICMC 2008)*, 2008.

[12] A. Agostini and D. Ghisi, "bach: An environment for computer-aided composition in max," in *Proc. Int. Computer Music Conf. (ICMC 2012)*, Ljubljana, Slovenia, 2012.

[13] C. Hope and L. Vickery, "The decibel scoreplayer - a digital tool for reading graphic notation," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2015*, Paris, France, 2015, pp. 58–69.

[14] R. Constanzo, "dfscore," Online, 2016, available: http://www.rodrigoconstanzo.com/dfscore/ (accessed Jan. 7, 2026).

[15] R. Burloiu, A. Cont, and A. Poncelet, "A visual framework for dynamic mixed music notation," *J. New Music Research*, vol. 46, no. 4, pp. 321–332, 2017.

[16] J. MacCallum, R. Gottfried, I. Rostovtsev, J. Bresson, and A. Freed, "Dynamic message-oriented middleware with open sound control and odot," in *Proc. Int. Computer Music Conf. (ICMC 2015)*, 2015.

[17] P. González-Fernández, F. P. Olives, E. R. Bertram, T. Noll, and P. beim Graben, "Quantum automation of playful heterophonic processes," in *Proc. Int. Computer Music Conf. (ICMC 2025)*, Boston, MA, USA, 2025, pp. 460–466.

[18] T. Noll, F. P. Olives, P. González-Fernández, and P. beim Graben, "Exploring quantum tonality," in *3rd International Symposium on Quantum Computing and Musical Creativity (ISQCMC '25)*, Palermo, Oct. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.18203933

[19] T. Noll, P. González-Fernández, and R. Yamada, "The quartet: Embodying quantum theory as an ensemble," 2026, manuscript submitted for publication.