DISTRIBUTED NOTATION IN THE BROWSER, AN OVERVIEW

Jonathan Bell
Aix Marseille Univ, CNRS, PRISM
Perception, Representations, Image, Sound, Music
Marseille, France
belljonathan50@gmail.com

ABSTRACT

This paper endeavours to discuss a few available technologies for musical notation, from a user's point of view, focusing on "distributed notation": musical notation rendered in real time to multiple devices simultaneously. The rapid evolution of browsers and the cheaper cost of heterogeneous (cross-platform) systems inclines us to narrow down the scope to browser-based solutions. The overview recalls a few key concepts of the JavaScript ecosystem, typically addressed to composers with little background in web development. The survey then compares three frameworks: INScore (INScoreWeb), DRAWSOCKET, and SmartVox, focusing on their respective approach to software architecture/implementation, as well as their "higher" user level. After highlighting the convergence points between INScore and DRAWSOCKET - but also their own specificity, such as INScore's time model, or DRAWSOCKET's API, the paper concludes with a case study: the Tenor 21 choral concert.

1. INTRODUCTION: MUSICAL NOTATION IN THE DIGITAL AGE

1.1 Animated Notation

Numerous Australian composers/academics have composed with and written about digital forms of notation: Vickery sees in animated screen-based notation '*an important solution to visualising a range of musical phenomena and techniques including continuous parametrical changes, synchronisation with prerecorded audio or live processing, and nonlinear formal organisation*' [1]. Hope, citing Winkler, sees animated notation as '*a third way between improvisation and fixed scores*' [2]. In such settings, Wyatt questions the role of the conductor [3] when for instance the timing is dictated by the advancement of a cursor on the page. Kim-Boyle, finally, pioneers research at the intersection between notation and immersive *augmented reality* technologies [4] (see section 4.1.1).

These approaches to notation present a drastic change in the organisation of musical time, which otherwise still follows today principles inherited from the theory accompanying the motets of the *ars nova*. [1] To paraphrase Vickery, animated notation simplifies the synchronization of human performers to multimedia, as well as the display of generative or interactive scores in which the work can render a different content each time [5], or adapt to the performance of the performer [6]. Among these technologies, three frameworks allow for real-time score display in the browser.

- INScore, which was operational as early as 2012 [7], releases in 2020 its web version (INScoreWeb).
- DRAWSOCKET [8, 9], which elaborates on its authors' former (albeit still actively developed) notational projects [10, 11, 12, 13]
- SmartVox, best described as a web-based distributed media player [14].

Whilst SmartVox is dedicated to a very specific task (the display and synchronisation of mp4 video files), INScore and DRAWSOCKET - which also both supports videos - share with the Decibel Score Player (see [15], 2. The Canvas scoring mode) the ability to render in real time SVG [2] drawing commands via Open Sound Control (OSC) [16] messaging in distributed setups (i.e. on multiple devices).

1.2 Web-Based cases

The present study strictly observes browser based solutions which is why the landmark Decibel Score Player [15], a native iOS application, is not discussed here. Network communication is evidently simplified by web/browser-based technologies, however the Decibel Score Player and a few other projects fully support this key feature of distributed notation, such as: Pedro Louzeiro's *Comprovisador*, which distributes notation in real-time to several clients using *bach* [17] in Max/MSP, communicating through UDP (see section 2.1.3), or Slavko Zagorac's ZScore [18], which is built on the top of the native version of INScore [7].

After a brief overview of the web technologies most commonly used by these notational systems, the paper introduces an non-exhaustive list of recent frameworks/pieces involving browser-based musical notation (Section 3.1), in order to compare a few of their general characteristics (Section 3.2). The paper then mainly focuses on DRAWSOCKET,

[1] Attributed to Philippe de Vitry, the division of large temporal units (*tempus*) into smaller one (*prolatio*), roughly corresponds to today's time signatures, see for instance: https://en.wikipedia.org/wiki/Prolation

[2] https://en.wikipedia.org/wiki/Scalable_Vector_Graphics

INScore and SmartVox, which are best known by the author.

2. OVERVIEW OF ASSOCIATED TECHNOLOGIES

2.1 Communication Protocols

2.1.1 HTTP

Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML. Riding on the top of TCP, it was designed for communication between web browsers and web servers.

2.1.2 WebSocket

WebSocket is distinct from HTTP. WebSocket is a communications protocol that provides full-duplex communication channels over a single TCP connection. It has become the *de facto* standard for real-time interaction with the browser.

2.1.3 TCP-UDP

TCP is a connection oriented protocol. UDP is a connection less protocol. As TCP provides error checking support and also guarantees delivery of data to the destination router this make it more reliable as compared to UDP. On other hand UDP is faster and more efficient than TCP.

2.1.4 OSC

OSC [16] is a content format developed at CNMAT by Adrian Freed and Matt Wright. Typically intended for sharing music performance data (gestures, parameters and note sequences) between musical instruments either via UDP (most common scenario) or TCP.

2.1.5 odot

OSC developments at CNMAT later led to odot [19] which allows to label data with human readable text (associative arrays). [3] Although discussed in greater detail in later sections, Figure 2 shows the translation operated by DRAWSOCKET, from an OSC bundle (odot) into JSON.

2.2 JavaScript

Javascript is commonly referred to as the language of the web. Originally client-side, Javascript engines are now embedded in some servers, usually via node.js. The most comprehensive documentation about the language can be found at developer.mozilla.org.

2.2.1 JSON

JavaScript's native format, JSON - JavaScript Object Notation - has eclipsed XML and thus became the most ubiquitous data exchange format for any publicly available web service. JSON supports associative arrays as well as ordered lists of values (arrays).

[3] An associative array, map, symbol table, or dictionary is an abstract data type composed of a collection of (key, value) pairs, such that each possible key appears at most once in the collection.

2.2.2 Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js owes its success to its module exchange platform NPM [4], as well as to the convenience of writing the client and the server in the same language.

2.2.3 Node for Max

The release of Max 8 in 2018 featured a new Javascript support with Node for Max. The Max API module helps interaction and communication with Max from within the Node context. [5]

2.2.4 Frameworks

Node.js is low level, which is why most applications use frameworks to deal with common server features. Routing, most importantly, is used to divert users to different parts of the web applications based on the request made. The Express framework has emerged as one of the most popular framework among Electron, koa, meteor and vue.js to name a few.

2.2.5 WebAssembly

WebAssembly (abbreviated Wasm) is a new type of code that can be run in modern web browsers. Providing languages such as C/C++, C# and Rust with a compilation target so that they can run on the web, the main goal of WebAssembly is to enable high-performance applications on web pages designed to run alongside JavaScript.

3. AN OVERVIEW OF WEB TECHNOLOGIES FOR DISTRIBUTED NOTATION

'*Distributing Music Scores to Mobile Platforms and to the Internet*' [20], an idea idea expressed by Fober in 2015, enjoys a growing interest today, as exemplified by two major frameworks (see table 1) as well as multiple independent initiatives by individual composers/developers (see table 2), all briefly described in the list below.

3.1 Description

3.1.1 INScore

INScore [7] is an environment for the design of augmented interactive music scores, open to unconventional uses of music notation and representation, including real-time symbolic notation capabilities. It can be controlled in real-time using Open Sound Control [OSC] messages as well as using an OSC based scripting language, that allows designing scores in a modular and incremental way. INScore supports extended music scores, combining symbolic notation with arbitrary graphic objects. All the elements of a score (including purely graphical elements) have a temporal dimension (date, duration and tempo) and can be manipulated both in the graphic and time space. The INScore engine is based on a MVC architecture. The abstract

[4] https://www.npmjs.com/

[5] https://docs.cycling74.com/nodeformax/api/module-max-api.html

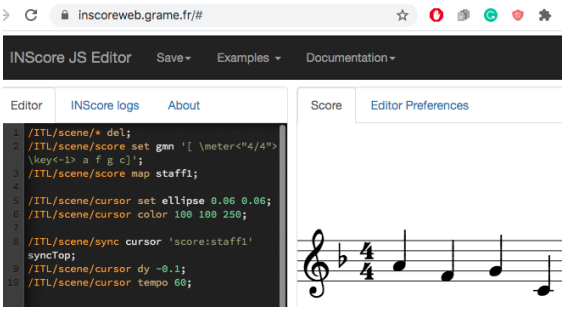

Figure 1. The INScore IDE is available online at : https://inscoreweb.grame.fr/.

model is designed in C++ and can be deployed on any platform (Windows, MacOS, iOs, Android) including the Web where it is compiled as a WebAssembly module (see Section 2.2.5. INScore Web now has an IDE (Integrated Development Environment) available online (see Figure 1, and the INScore Web package will be available on NPM in 2021.

3.1.2 DRAWSOCKET

DRAWSOCKET is best described in [8, 9], which is why only references tracing its genealogy are provided here. The roots of the DRAWSOCKET project can be traced back in Hajdu's quintet.net [11], already a landmark piece of software in the realms of distributed notation and networked music performances when the internet was still a young technology. More recent works by Gottfried on 'SVG to OSC transcoding' [13] also reveal some of its premises.

Odot (OSC) Bundle :	JSON object (or Python dictionary)

/violin : {
/key : "svg",
/val : {
/new : "circle",
/id : "bob",
/cx : 100,
/cy : 100,
/r : 20,
/fill : "green"
}
}
	{
	"violin" : {
	"key" : "svg",
	"val" : {
	"new" :"circle,
	"id" : "bob",
	"cx" : 100,
	"cy" : 100,
	"r" : 20,
	"fill" : "green"
	}
	}
	}

Figure 2. Comparison between an odot bundle (OSC), and a JSON object, following DRAWSOCKET's API syntax.

The transcoding of OSC notational commands to JSON (see Figure 2) describes it well, and highlights its potential for versatile real-time notational purposes in the browser. Taking advantage of the odot [19] library's ability to dynamically format and label OSC bundles, the node.js framework is embedded in Max (via Node for Max, see section 2.2.3), but can also run independently from the Max environment [6] . DRAWSOCKET provides the user with an API [7] dedicated to real-time communication with connected clients. As shown in Figure 2, the top level router sends the bundle to a specific client (here the violin). Replacing /violon by /* would send the bundle to all. Then each keyword (/key or "key" in Figure 2) obeys a slightly different - although consistent - syntax : "svg" will draw an svg on the page, "tween" will animate the svg according to its target /id value (in figure 2 the target svg's id is "bob"), "pdf", "sound" and "file" will load the corresponding files (e.g. pdf, mp3 or JSON). The Elbe Tunnel's scores repository provides an apt example for observing DRAWSOCKET in action. [8]

3.1.3 SmartVox

Developed at IRCAM in 2016 in the SoundWorks framework [21] of the CoSiMa project (ANR-13-CORD-0010), SmartVox [14] consists of distributing and synchronizing mp4 audiovisual scores on the browsers of the performers' smartphones (typically on a network heterogeneous local i.e. on different OS or cross-platform), to help them sing in polyphony, in site specific settings (e.g. moving around the audience), and in a spectral language (i.e. microtonal). Written in Javascript, SmartVox uses the native JSON format to assign each singer a unique file (see Figure the script example below).

```
const score = {
  duration: 20 * 60, // seconds

  // define the different parts
  parts: {
   'soprano-1': {
      file: 'videos/soprano-1.mp4',
    },
    'soprano-2': {
      file: 'videos/soprano-2.mp4',
    },
    // ...
  },

  // define the different sections
  sections: {
    alpha: {
      time: 0,
      label: 'First_section',
    },
    beta: {
      time: 117,
      label: 'Second_section',
    },
    // ...
  },
};
```

3.1.4 Prolonged Into the Latent (PITL)

Amongst an ever-increasing number of solutions for web-based synchronised scores, Justin Yang's *Prolonged into the latent (PITL)* [9] provides a good entry point for considering how two types of clients (the conductor [10] and one of the singers [11]) can communicate via WebSockets (see Section 2.1.2). Yang's notational interface is reminiscent of the *Guitar Hero* video game.

[6] See https://drawsocket.github.io/index.html
[7] See: https://drawsocket.github.io/api.html
[8] https://quintetnet.hfmt-hamburg.de/tunnel_webviewer/index.html
[9] The code is available at: https://github.com/elosine/prolonged_into_the_latent
[10] The conductor's interface can be accessed via the following url: https://pitl.justinyang.net/?parts=0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15&controls=yes
[11] The singer's interface can be accessed via the following url: https://pitl.justinyang.net/?parts=0, pressing 'start' on the conductor will cause the singer's part to start playing. Hitting the ready button on the singer's interface will

3.1.5 John

Goudard's *John, the semi-conductor: A tool for comprovisation* [12] [22], is a distributed notation software '*designed to help collective free improvisation*'. The term *Comprovisation* refers to the composer and scholar Sandeep Bhagwati [23].

3.1.6 Anna und Marie (A&M)

Pirchner [13] [24] developed a real-time score-system for the composition *Anna & Marie (A&M)* by Marko Ciciliani. SuperCollider sends OSC messages to the dedicated score system generating symbols and playing instructions for each performer, rendering them on tablet displays.

3.2 Comparison

The six aforementioned frameworks will now be compared according to the following categories : 1) Which type of server does the setup rely on. 2) Client side, does the framework target the web or a specific operating system 3) Does it have WebSocket support (see Section 2.1.2). 4) Does it use a framework such as Express (see Section 2.2.4). 5) Is it a framework, or was the architecture build for a single piece 6) Does it support sound. 7) Does it support tradition notation (abbreviated CMN for Common Music Notation) 8) Does the framework rely on a shared clock/which library does it use from this 9) Does it support Scalable Vector Graphics. 10) Does it use pre-existing graphic libraries.

	INScore	**Drawsocket**
Server	any server	node.js
Web/Native	any OS	Web
Websockets	ws	ws
uses a framework	no	express
is a framework	yes	yes
Sound	faust	tone.js
CMN	Guido	Bravura
Sync	native	timesync
SVG support	yes	yes
Graph. lib., animation	qt	tween d3js

Table 1. Comparison of architectural characteristics of two major frameworks for distributed notation.

3.2.1 node.js/WebSocket

Most importantly, the first two rows of the tables shed light on the strong presence of node.js on the server side, together with WebSockets, today well-known technologies for implementing real-time communications in the modern web. The case of INScore is different in the sense that it privileges self-containedness over server/client architecture: the page can therefore be delivered by servers of any kind such as python, Apache, or node.js.

The second row of Table 2 shows that three technologies use socket.io, [14] a JavaScript library built on top of WebSocket. DRAWSOCKET and some parts of PITL use the express.js framework, *SmartVox* uses *SoundWorks*[21].

3.2.2 Sound and Common Music Notation (CMN)

Regarding their sonic capacities, INScore has the advantage to embed *faust* [25], and therefore promises interesting deployments of DSP and sound synthesis in the realm of *musique mixte*. *PITL* uses the web audio API and DRAWSOCKET uses tone.js, a simple library built on the top of it. For traditional notation (CMN), INScore natively supports GUIDO. [15]

3.2.3 Synchronisation

DRAWSOCKET and PITL show that *timesync* is today the most commonly used library to, for instance, make sure a message arrives to all clients exactly at the same time, according to a shared clock (Smartvox uses an IRCAM library [26], based on a similar concept).

3.2.4 Graphics and Animation

Whilst most of the applications discussed so far use SVG (Scalable Vector Graphics being the prominent solution to define graphics for the web), only INScore and DRAWSOCKET, support SVG rendering in real time.

DRAWSOCKET, as its names states, *draws* SVG on a page, without an intermediary library, but uses GSAP-tween for animation [16] , which can be conceived as 'high-performance property setter'. Regarding the use of external libraries for graphic renderings, no clear convergeance can be found across the six frameworks considered here.

	PITL	**A&M**	**John**	**SmartVox**
Server	node.js	node.js	node.js	node.js
Web/Native	Web	Web	Web	Web
Websockets	socket.io	socket.io	∅	socket.io
uses a fr.	express	electron	Meteor	SoundWorks
is a fr.	no	no	yes	yes
Sound	webaudio	∅	∅	(mp4)
CMN	∅	∅	∅	∅
Sync	timesync	∅	∅	@ircam
SVG support	yes	yes	yes	no
Graph. lib.	Three.js	snap.svg	d3js	∅

Table 2. Comparison of architectural characteristics of a few browser-based frameworks for distributed notation.

After this rapid technical overview of the landscape of real-time/distributed notation, technical considerations as well as user experiences with INScoreWeb, SmartVox and DRAWSOCKET will be discussed in greater detail, in order to highlight what brings them closer and where they diverge.

4. COMPARATIVE STUDY

When compared two by two, the three aforementioned frameworks reveal a few striking similarities (see Section 4.3/Figure 3), but also raise questions when they approach the same domains differently.

[12] Code available at: https://github.com/vincentgoudard/ReactiveJohn

[13] See online repository : https://github.com/asa-nerd/Anna-und-Marie

[14] Socket.IO primarily uses the WebSocket protocol with polling as a fallback option, while providing the same interface. Although it can be used as simply a wrapper for WebSocket, it provides many more features, including broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O.

[15] https://guidodoc.grame.fr/#welcome-to-guido

[16] Available at: https://greensock.com/docs/v3/GSAP/Tween

4.1 INSCORE and SmartVox

4.1.1 Augmented Reality

The author has documented some of his own artistic works in the realm of AR notation with SmartVox [27, 28, 29]. Emerging works such as [4, 30] let us envisage rapid developments in this field, in which the prescription of a gesture in space should reveal representations far more intuitive (often called prescriptive, or tablature-like, as opposed to descriptive i.e. relying on an abstract system such as the 5-line staff) than those used on paper for centuries, or those developed on animated screens since a few decades.

Although achieved by [31], in which four performers were guided by Hololens, the rapid evolution and ephemerality of such technology lets us favour cheaper smartphone solutions. So far the straightforward method used by the author consisted a simple display above the head of the performer. With such cheap setups, however, holographic display - which requires a different image for each eye - often failed to provide a comfortable display due to calibration issues linked to the size of the performer's phone and length of his/her inter-pupillary distance.

INScore can run as a native Android application, which has given evidence of very promising results on EPSON bt-350 glasses in this domain. INScore's forwarding mechanism [17] allows for the real-time transfer from one INScore instance (on a laptop) to another (on glasses), by writing the following script, in which '$glasses' corresponds to the ip address of the target device.

```
glasses = "192.168.0.26:7000";
/ITL forward $glasses;
```

This proved convenient for debugging purposes, and revealed a comfortable display for the user. The native Android version of INScore enjoyed stable results. The downside of EPSON bt-350 glasses, for browser-based tests, is that it currently fail to load html pages served by DRAWSOCKET, INScoreWeb, or SmartVox.

4.1.2 Sound

Sound may not seem a central feature for softwares dedicated to musical notation. However, it has been demonstrated in [32, 33, 34] that many fruitful artistic works rely on the use of sounds or auditory signals as a score - e.g. a guide track for a singer rather a sound file to be played through loudspeakers. SmartVox, essentially a distributed mp4 player, takes advantage of the HTML video tag, whose embedded media supports audio as well as video.

INScore's browser implementation [35], thanks to the recent release of the Faust NPM package [18] (Faust compiled as a WASM library), currently investigates the possibilities of exploiting the DSP capacities of the web page rendering the score. The feature will therefore allow (among other possibilities) a precise synchronisation between electronic transformations and the score, e.g. the opening of the browser's microphone when the cursor hits a certain note.

[17] Described here: https://inscoredoc.grame.fr/refs/10-forwarding/

[18] https://www.npmjs.com/package/@grame/libfaust

4.2 DRAWSOCKET and SmartVox: cache memory and delay management

4.2.1 Join the performance at any time

Compared to native applications, web pages are fragile, the following issues in particular worth be considered: 1) Web pages can sometimes load improperly due to network issues 2) Some features may not function depending on which browser and which OS is being used 3) Smartphones' defaults behaviours (mute, night shift, sleep mode...), or unintended user interactions (such as swipes, clicks or earphone pluging) may cause perilous effects in performance situations.

In such cases, reloading if often required. Then problems may arise if something goes wrong in the middle of a musical performance, and if the current state of the application is not stored in the cache (e.g. which bar in the composition timeline are we in 'now'?). In DRAWSOCKET, the caching system stores automatically all the drawing commands, so that refreshing the page client side keeps trace of all the drawing commands since the server started, or since the page was cleared. When a client is late, a mechanism allows it to catch up delay when a message is received too late by a client [19]. More precisely, for tween animations specifically (see footnote No 15), with the 'cmd' function, DRAWSOCKET will check to see the difference between the start time and the current time (according to the shared clock discussed in Section 3.2.3) and jump ahead if it is late. [20]

One of the strength of SmartVox consists of its ability to update the 'currentTime' (the instant currently displayed in the video) on each server tick: when clients periodically query the shared clock to check whether the drift is not too important [21]. This feature proved to be very robust in several performances [22], if for instance, in the middle of a piece, a problem occurs, and a performer needs to refresh his/her page, or if he/she was not ready when the piece started.

4.2.2 Scheduler

The default behaviour of DRAWSOCKET privileges instant responsiveness over synchronisation. If for instance a sound is triggered to all clients, this sound will be played a soon as possible. Rather than delaying messages of a given value to make sure every one receives it at the same time, when using the 'cmd' message, the sound it will jump ahead if it is late. The mechanism is comparable to the one explained earlier with tweens (cf. footnote 20).

Each incoming message is automatically time-tagged according to a shared clock. While writing this article, two different messages ('del' and 'schedule') are being tested, both locally and in remote setup. 'schedule' delays a mes-

[19] See the source code :https://github.com/HfMT-ZM4/drawsocket /blob/master/code/node/lib/drawsocket-client.js, line 947-1069 and 1567-1613)

[20] Reloading the page in the middle of a tween animation: https://youtu.be/2ahjjbS5s2U

[21] See the source code here: https://github.com/belljonathan50/SmartVox0.1 /blob/master/src/client/player/PlayerExperience.js, line 153).

[22] e.g. Deliciae, Common Ground, Le temps des nuages, SmartVox... All described in former TENOR publications by the same author.

sage of a given value according to the clock sync offset, while 'del' simply delays it. More specifically, these two messages are part of an in-development 'event' [23] processing system, that aims to provide a generalized API for synchronizing events.

4.3 INSCORE - DRAWSOCKET, similarities

In order to show how, fundamentally, INScore and DRAWSOCKET obey the same kind of OSC-driven drawing commands, both of the following scripts draw a line or rectangle named "cursor", at the top left of the violin part. In INScore the origin is in the center which is why x and y are negative (scaled between -1. and 1.). In DRAWSOCKET the origin is at the top left, and counts in pixels (see Figure 3, the first 3 line in the INScore script, middle column in DRAWSOCKET). To animate the cursor object and thus move it across the score, however, INScore and DRAWSOCKET use different methods, (see Figure 3, line 4 until the end in the INScore script, right column in DRAWSOCKET), which will be detailed in more details in the following section.

INScore & INScore WEB	DRAWSOCKET	

## cursor creation: 	
 	
/ITL/scene/cursor set rect 0.01 0.2;
/ITL/scene/cursor color black;
 	
## creation of a linear trajectory (line): 	
/ITL/scene/line set line wa 1 0;
/ITL/scene/line x -0.5; ## towards left
/ITL/scene/line y -0.7; ## upwards
/ITL/scene/line duration 1; ## 4 seconds
 	
## synchronization of the cursor to the line: 	
/ITL/scene/cursor tempo 60 ;
/ITL/scene/sync cursor line syncFrame;
	"// cursor creation"
	
	/violin : {
	/key : "svg",
	/val : {
	/new : "line",
	/id : "cursor",
	/x1 : 200,
	/x2 : 200,
	/y1 : 40,
	/y2 : 80,
	/style : {
	/stroke : "black"}
	}
	}
		"// definition of a displacement of the cursor of 200 pixels towards the right-hand side"
		
		/violin : {
		/key : "tween",
		/val : {
		/id : "score-tween",
		/target : "#cursor",
		/dur : 5,
		/vars : {
		/x : 200,
		/paused : "true"
		}
		}
		}

Figure 3. Animating a cursor in INScore and DRAWSOCKET.

4.4 INSCORE - DRAWSOCKET, a significant difference in animation design

4.4.1 INScore Time Model

The description of time in INScore [36] shares with two other French projects (Antescofo [37, 38] and Iscore [39]) similar concerns about technology's ability to handle both continuous and event-driven time. INScore objects therefore have a duration and a date in order to be then synchronized graphically according to their temporal relation, which makes it possible for example to "monitor" an event to trigger another (e.g. trigger a page turn after a cursor the cursor finished crossing the staff. . .). In the example above (see Figure 3) the cursor is synchronised (according to a given tempo - 60) to a line (a segment of a certain length in the graphic space, and of a given duration).

In INScore, synchronising an object (a) to another (b) has a very peculiar meaning, which may be understood as 'making x adopt the spatial properties of y, function of time'. Although this is only one way to understand them, but as useful entry point example, INScore graphical objects may be interpreted as belonging to two different categories : cursors (a) and trajectories (b), or, in other terms players/pointers (a) and score (b). Typically, the cursor is 'synchronised' to a given trajectory (master). Just like a traditional (fixed) score, 'b' can be executed at a slightly different speed each time by its interpreter (a), we therefore attribute a fixed date and duration the score b, and a tempo to the cursor a (an example in *absolute time* is provided in section 5.5, *absolute time* constrats with *musical time* which is tempo-relative).

We find interesting to recall here how various improvements of INScore eventually led to the new specification of a 'tempo' attribute (see [34], end of the introduction), giving evidence of the complex demands of musical time. In *Perspective Temporelles* [24] , observing how the cursor's speed changes at the very beginning might be an apt exemple for illustrating how the time to graphic relationship often needs refined adjustments ([3], section 2.1, time to graphic relation). Indeed, in the example mentioned in Footnote No 24, the horizontal trajectory of the cursor is divided in two segments [470 , 540[and [540, 2880[, [25] each being given a corresponding duration, hence the perceived effect of an acceleration of the cursor.

4.4.2 Animation in DRAWSOCKET

In contrast, DRAWSOCKET offers a simple animation solution, the "tween", from the GSAP library - a standard for javascript animation in HTML5. The animation is defined according to the time and the graphic space to be traversed. The tween approach is easier to handle that INScore for the simple cases, but may on the other hand encounter limitations when the temporal unfolding is not linear with respect to the graphic space. To handle such cases, DRAWSOCKET's tween implementation supports 'multi-segment tween timelines' (accessed from DRAWSOCKET's helpfile, tween animation tab). [26]

5. CASE STUDY: TENOR 21 CHOIR CONCERT

This article is written during a time when the world is profoundly impacted by the aftermath of the Covid-19. Some constituted vocal ensembles have had to endure a whole year without any rehearsals. Contemporary musical practice is suddenly forced to operate massively within a field that Hajdu [11] had investigated since the early days of the internet, and which, at the time, was anything but easy to realise : 'Devising a network performance environment, such as my Quintet.net, is probably among the most demanding tasks a composer or visual artist can face today' [11]. Since realtime audio transmission over the internet was unthinkable at the turn of the millenium, Hajdu had envisaged a system (quintet.net) in which the music played by the instrumentalists were recorded and subsequently encoded to midi, for distribution over the network. As for-

[23] See the API, in the 'event' keyword section : https://drawsocket.github.io/api.html#event

[24] The piece is available at http://berio.grame.fr/perspectives/

[25] For an explanation of the Segment definition syntax, see: https://inscoredoc.grame.fr/refs/12-mapping/#segments-definitions

[26] Many built-in easing functions in GSAP do non-linear movement, and also it's possible to write one's own timer functions with GSAP https://greensock.com/docs/v3/Eases.

mulated in the title of his article 'Embodiment and disembodiment in networked music performance' [40], Hajdu has also anticipated from an early stage the many possible shortcomings (technical as well as artistic) of performances in which the players are located at a long distance from between each other. Another well-know pre-Covid limitation of network music performance was induced by the fact that network delay prevented musicians from rhythmically responding to each other [41].

As a response to the crisis, and under Hajdu's initiative, a remote choral concert was organised at the Hambourg Horschule for Tenor 2021, in which each singer was provided which a low latency audio kit (Rasbperry Pi 4 + microphone + Soundcard), thanks to the effort of Jacob Sello configuring twenty Raspberry Pi-embedded JackTrip clients [42], together with 2 iPads (one for zoom sessions, and the other for DRAWSOCKET distributed notation).

Five pieces were rehearsed for the concert, all using DRAWSOCKET in various ways, together with low latency audio. This presented the obvious advantage that the singers could all rehearse together from home, and the composers located in four different countries could also attend the rehearsals.

A DRAWSOCKET feature which proved most helpful here was its ability to dynamically load different pieces of the concert, so that the singers (clients) didn't need to do anything else than join their allocated url at the start of the concert.

5.1 Anders Lind: The Max Maestro

The interface for this piece is originally a Max patch, and was entirely re-written in DRAWSOCKET, so that the composer could remote control the live-generated notation from Sweden while the choir was rehearsing in Hamburg. [27]

5.2 Justin Yang: Prolonged into the latent (PITL

As seen earlier in Table 2, *Prolonged into the latent (PITL)* has its own environment, and DRAWSOCKET was therefore used here only to encapsulate Justin Yang's website within an iframe.

```
/bas4 : {
  /key : "html",
  /val : {
    /new : "iframe",
    /parent : "forms",
    /id : "iframe_ex",
    /style : {
      /position : "absolute",
      /top : "0px",
      /left : "0px",
      /width : "100vw",
      /height : "100vh"
    },
    /src :
"https://pitl.justinyang.net/?parts=0"
  }
}
```

DRAWSOCKET redirects here to the lowest voice of the composition (bass 4, accessed via the DRAWSOCKET url concatenated with '/bas4') to the corresponding part of Justin Yang's website. [28]

[27] Original work by Anders Lind: https://youtu.be/4iePLi5uQzU. Version ported to DRAWSOCKET by Jonathan Bell : https://youtu.be/sdSyHIbK5FY

[28] Available at: https://pitl.justinyang.net/?parts=0&controls=yes

5.3 Richard Hoadley: Unthinking Things

Unthinking Things was originally written in INScore. The algorithmic work was composed by Hoadley using SuperCollider sending control messages to INScore [43]. The port to DRAWSOCKET for the concert consisted of a video of the INScore-generated piece, served and synchronised by DRAWSOCKET. [29]

5.4 Jonathan Bell: Common Ground

Common Ground [28] is originally written in bach [17], and most specifically in its most recent *bell* textual extension [44, 45]. The original performance involved singers dancing in an immersive space [28], with scores distributed and synchronised by SmartVox embarqued on a Raspberry Pi. Like for Hoadley, the final performance consisted of videos synchronised via DRAWSOCKET. [30]

5.5 Palestrina: O crux Ave

The Palestrina work was the only one whose language is based on a regular - albeit very slow - pulse, which raised questions mentioned earlier [3] about the role of the conductor when the pulse can be conveyed via animated notation means. The rehearsals allowed iterative attempts with pulse-based animations [31] and contrasting approaches with scrolling cursors realised with INScore, with which the mapping between time and the pixel-accurate cursor position on the screen can be notated with great precision: in the following example for instance, the cursor travels the distance between x1 (208) and x2 (249) in one second (t2 - t1), then the distance between x2 (251) and x3 (305) in 1 second (t3 - t2). [32]

```
 # x1   x2    y1    y2       t1          t2
([208, 249[ [93, 394[) ([0:0:0, 0:1:00[)
 # x2   x3    y2    y3       t2          t3
([251, 305[ [97, 394[) ([0:1:00, 0:2:00[)
```

The perspective envisaged for future attempts will consist of capturing the conductors gesture with gesture follower technology [46] to overcome the limitations caused by the current lag of today's video conferencing platforms (such as zoom), which currently makes conducting impractical.

6. CONCLUSION

With the present survey, the author hopes to have shed light on the emerging field of distributed notation in the browser. With the DRAWSOCKET API and the synchronisation capabilities of INScore, if we think of the scaling capacities of such technologies (the Hamburg St.Pauli Elbe Tunnel performance [9] involved a 144 musicians [33], Le temps des nuages was premiere with 80 singers [34]), or the unforeseable perfoming situations these may lead to when

[29] The score is available at : https://youtu.be/gLWvjR8vPHw

[30] A recording is available at : https://youtu.be/ZrLgbBw4xfU

[31] A version of the piece animated via the GSAP-tween library in DRAWSOCKET is available here: https://youtu.be/3SS9Cb0AtU0

[32] For an explanation of the Segment definition syntax, see: https://inscoredoc.grame.fr/refs/12-mapping/#segments-definitions

[33] Score: https://quintetnet.hfmt-hamburg.de/tunnel_webviewer/index.html Performance: https://youtu.be/cdnA_ZijYUI

[34] Recording with the score: https://youtu.be/SyFdR2HiF00 Performance: https://youtu.be/7j2_D-nQAHY?t=6424

combined with increasingly accessible AR technologies, we hope the cases discussed here will prompt more composers to investigate this exciting field.

Acknowledgments

I am would like to thank Dominique Fober, for his generous time ans support, as well as the Hamburg team for the Tenor 21 choral concert experience.

7. REFERENCES

[1] L. Vickery, "The limitations of representing sound and notation on screen," *Organised Sound*, vol. 19, no. 3, pp. 215–227, 2014.

[2] C. Hope, "Electronic scores for music: The possibilities of animated notation," *Computer Music Journal*, vol. 41, no. 3, pp. 21–35, 2017.

[3] A. Wyatt and C. Hope, "Conducting animated notation: Is it necessary?" in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'20*, Hamburg, Germany, 2020, pp. 169–174.

[4] D. Kim-Boyle, "3d notations and the immersive score," *Leonardo Music Journal*, vol. 29, pp. 39–41, 2019.

[5] J. Freeman, "Extreme sight-reading, mediated expression, and audience participation: Real-time music notation in live performance," *Computer Music Journal*, vol. 32, pp. 25–41, 09 2008.

[6] A. Cont, "ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music." in *International Computer Music Conference (ICMC)*, Belfast, Ireland, Aug. 2008, pp. 33–40.

[7] D. Fober, Y. Orlarey, and S. Letz, "INScore - An Environment for the Design of Live Music Scores," in *Linux Audio Conference*, Stanford, United States, 2012, pp. 47–54.

[8] R. Gottfried and G. Hajdu, "Drawsocket: A browser based system for networked score display"," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, Melbourne, Australia, 2019, pp. 15–25.

[9] G. Hajdu and R. Gottfried, "Networked music performance in the old elbe tunnel," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*. Melbourne, Australia: Monash University, 2019, pp. 55–60.

[10] G. Hajdu and N. Didkovsky, "Maxscore: Recent developments," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, Montreal, Canada, 2018, pp. 138–146.

[11] G. Hajdu, "Quintet.net: An environment for composing and performing music on the internet," *Leonardo*, vol. 38, no. 1, pp. 23–30, 2005.

[12] R. Gottfried and J. Bresson, "Symbolist: An open authoring environment for user-defined symbolic notation," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, Montreal, Canada, 2018, pp. 111–118.

[13] R. Gottfried, "Svg to osc transcoding as a platform for notational praxis and electronic performance," in *Proceedings of the First International Conference on Technologies for Music Notation and Representation – TENOR'15*, Paris, France, 2015, pp. 154–161.

[14] J. Bell and B. Matuszewski, "Smartvox. a web-based distributed media player as notation tool for choral practices," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'17*, A Coruña, Spain, 2017, pp. 99–104.

[15] A. Wyatt, L. Vickery, and S. James, "Unlocking the decibel scoreplayer," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, Melbourne, Australia, 2019, pp. 61–68.

[16] M. Wright, "Open sound control: An enabling technology for musical networking," *Org. Sound*, vol. 10, no. 3, pp. 193–200, Dec. 2005.

[17] A. Agostini and D. Ghisi, "A max library for musical notation and computer-aided composition," in *Computer Music Journal*, 2015.

[18] S. Zagorac and M. Zbyszynski, "Networked comprovisation stratagies with zscore," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'20*, Hamburg, Germany, 2020, pp. 133–140.

[19] J. MacCallum, R. Gottfried, I. Rostovtsev, J. Bresson, and A. Freed, "Dynamic Message-Oriented Middleware with Open Sound Control and Odot," in *International Computer Music Conference*, ICMA, Ed. Denton, United States: University of North Texas, 2015.

[20] D. Fober, G. Gouilloux, Y. Orlarey, and S. Letz, "Distributing music scores to mobile platforms and to the internet using inscore," 2015.

[21] N. Schnell and S. Robaszkiewicz, "Soundworks – A playground for artists and developers to create collaborative mobile web performances," in *'Proceedings of the Web Audio Conference (WAC'15)*, Paris, France, 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01580797

[22] V. Goudard, "John, the semi-conductor: A tool for comprovisation," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, Montreal, Canada, 2018, pp. 43–49.

[23] S. Bhagwati, "Notational perspective and comprovisation," *Sound & Score. Essays on Sound, Score and Notation*, pp. 165–177, 2013.

[24] A. Pirchner, "Ergodic and emergent qualities of realtime scores. anna and marie and gamified audiovisual compositions," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'20*, Hamburg, Germany, 2020, pp. 189–197.

[25] Y. Orlarey, D. Fober, and S. Letz, "FAUST : an Efficient Functional Approach to DSP Programming," in *New computational paradigms for computer music*, E. D. FRANCE, Ed., 2009, pp. 65–96.

[26] J.-P. Lambert, S. Robaszkiewicz, and N. Schnell, "Synchronisation for Distributed Audio Rendering over Heterogeneous Devices, in HTML5," in *2nd Web Audio Conference*, ser. Proceedings of the 2nd Web Audio Conference (WAC-2016), Atlanta, GA, United States, Apr. 2016.

[27] J. Bell, "Networked Head-Mounted Displays for Animated Notation and Audio-Scores with SmartVox," NIME - New Interfaces for Musical Expression, Jun. 2019.

[28] J. Bell and A. Wyatt, "Common ground, music and movement directed by a raspberry pi," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'20*, Hamburg, Germany, 2020, pp. 198–204.

[29] J. Bell and B. Carey, "Animated notation, score distribution and ar-vr environments for spectral mimetic transfer in music composition," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, Melbourne, Australia, 2019, pp. 7–14.

[30] G. Santini, "Action scores and gesture-based notation in augmented reality," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'20*, Hamburg, Germany, 2020, pp. 84–90.

[31] D. Kim-Boyle and B. Carey, "Immersive scores on the hololens," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, Melbourne, Australia, 2019, pp. 1–6.

[32] J. Bell, "Audio-scores, a resource for composition and computer-aided performance," Ph.D. dissertation, Guildhall School of Music and Drama, 2016.

[33] S. Bhagwati, "Elaborate audio scores: Concepts, affordances and tools," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, Montreal, Canada, 2018, pp. 24–32.

[34] C. Sdraulig and C. Lortie, "Recent audio scores: Affordances and limitations," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, Melbourne, Australia, 2019, pp. 38–45.

[35] D. Fober, "A web based environment embedding signal processing in musical scores," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'21*. Hamburg, Germany: Hamburg University for Music and Theater, 2021.

[36] D. Fober, Y. Orlarey, and S. Letz, "INScore Time Model," in *International Computer Music Conference*, Shanghai, China, 2017, pp. 64–68.

[37] J.-L. Giavitto, J.-M. Echeveste, A. Cont, and P. Cuvillier, "Time, Timelines and Temporal Scopes in the Antescofo DSL v1.0," in *International Computer Music Conference (ICMC)*. ICMA, 2017.

[38] J.-L. Giavitto and J. Echeveste, "Real-Time Matching of Antescofo Temporal Patterns," in *PPDP 2014 - 16th International Symposium on Principles and Practice of Declarative Programming*. Canterbury, United Kingdom: ACM, Sep. 2014, pp. 93–104.

[39] A. Allombert, M. Desainte-Catherine, and G. Assayag, "Iscore: A system for writing interaction," in *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, ser. DIMEA '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 360–367.

[40] G. Hajdu, "Embodiment and disembodiment in networked music performance," in *Body, Sound and Space in Music and Beyond: Multimodal Explorations*. Taylor & Francis, 2017.

[41] C. Chafe, J.-P. Cáceres, and M. Gurevich, "Effect of temporal separation on synchronization in rhythmic performance," *Perception*, vol. 39, no. 7, pp. 982–992, 2010.

[42] C. Chafe, "JACKTRIP ON RASPBERRY PI," in *LAC - Linux Audio Conference*, 2019.

[43] R. Hoadley, "Homenaje a cervantes," in *TENOR'17*. A Coruña, Spain: Universidade da Coruña, 2017, pp. 229–238.

[44] J.-L. Giavitto and A. Agostini, "Bell, a textual language for the bach library," in *ICMC 2019 - International Computer Music Conference*, New York, United States, Jun. 2019.

[45] J. Bell, "Improvements in bach 0.8.1, a User's Perspective," in *SMC - Sound and Music Computing*, Turin, France, 2020.

[46] F. Bevilacqua, F. Guédy, E. Fléty, N. Leroy, and N. Schnell, "Wireless sensor interface and gesture-follower for music pedagogy," in *International Conference on New Interfaces for Musical Expression*, New York, United States, 2007, pp. 1–1.