1. INTRODUCTION

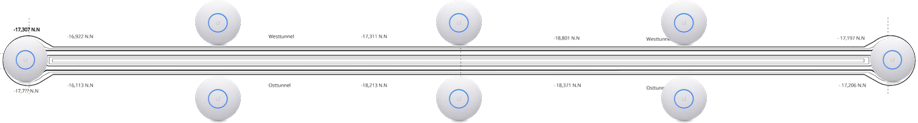
In a presentation about his piece Music for a Wilderness Lake (1979), R. Murray Schafer referred to it as a “music for a place (personal communication).” This music greatly depends on the *topology* of the environment it is performed in. While cases of topologically informed music (with static and/or moving musicians) have already existed in since the 1500’s (in his Memos Charles Ives tells the story of his father experimenting with marching bands walking towards each other and playing different tunes), it became more of a practice during the second part of the 20th century. Examples include Musik für ein Haus (Stockhausen, 1968), Eine Brise / Flüchtige Aktion für 111 Fahrräder (Kagel, 1996) and music by Alvin Curran who created music that was performed in the Sydney harbor or on the river Thames.

One of the difficulties of such a practice is maintaining synchronicity, as the participants either act on their own or execute scores with little coordination amongst each other due to lack of visual and auditory cues from a central agent / conductor. Such difficulties are aggravated if the location of the performance is a virtual one such as in networked music performance (NMP). Recent developments in digital technologies have leveraged some if not all of these difficulties by providing audiovisual tools. They are either capable of creating a shared space by low-latency streaming or the exchange of control messages. While audio/video streaming has yielded excellent results since the early 2000s (JackTrip), systems that feature music notation in networked environments have been rare and only become more widespread since about 2010. Quintet.net (1999), a networked multimedia performance environment developed by Georg Hajdu is an earlier example of software in which the interaction of musicians is facilitated by a visual notation layer. The European Bridges Ensemble was an ensemble for NMP whose members have been particularly focused on creating scores capable of capturing the specificities of the particular performance scenario. “Situative scoring” is a term coined by Sandeep Bhagwati describing scores that are defined as scores that deliver time- and context-sensitive score information to musicians when it becomes relevant…. Lately, reactive, interactive, locative scores have added new options to situative scoring.” (…)

In this paper we would like to describe the musical and technological developments towards a project dubbed *Symphony for a Tunnel* to be realized in Hamburg in May of 2019. The Old Elbe Tunnel is a remarkable landmark in the heart of the Hamburg port. Completed in 1911, it was considered a technological marvel at the time, connecting two neighborhoods below the Elbe river. Featuring two tubes for pedestrians, cyclists and automobiles which are being carried down / up by sizeable lifts to / from the bottom 24m beneath the surface, it is also an extraordinary place for performances. Its Jugendstil half cylinders form a resonant body in which the sound of a single instrument carries over large distances with relatively little decay (footnote: in an experiment featuring a violin playing scales at mezzo forte dynamics, it could be heard at 50m distance as if it was still be played right next to the observer).

The Stage\_2.0 grant within the Innovative Hochschule initiative of the Federal Ministry of Education and Research in Germany (BMBF) has finally laid the financial foundation for a musical project in the tunnel. The idea is to connect a large number of musicians via a network of connected devices delivering scores on time. We went through a number of scenarios until zooming in on the most practical solution: As the total circular length of the tunnel is about 860 m and an ideal spacing of individual musicians was determined by us to be around 5 to 6 m, it was a most welcome finding that dividing 864 by 6 yielded 144, a highly divisible number with technical and compositional repercussions. For instance, this number allowed us to define identical sub-groups consisting of 12 musicians each (Table of instruments) or to place eight access points at regular distances between musicians (Figure). An older, safer idea consisting of tablet computers connected to a wired Ethernet network was abandoned in favor of a Wi-Fi network as it would have required anywhere between 2.5 and 5 km of cables depending on the number of networking components involved. We finally decided on using iPads as the on-time delivery of scores was to be done via browsers, and Apple and its authorized resellers seemed most in tune with our needs [rephrase]. We are going to use the term *massive networked music performance* to denote performances that include more than 2 dozen nodes.

Another serendipitous finding was that when Rama Gottfried joined the Stage\_2.0 project in 2018, he had already been working on a node.js-based system for the Berlin Ensemble Mosaik and possessed the expertise to take on a project that would also take advantage of Cycling ’74’s recent effort to integrate node.js into their Max multimedia authoring environment.





2. FOUNDATIONS

A small number of software solutions are capable delivering scores in networked environments, most notably InScore, bach and MaxScore. MaxScore emerged from the ongoing effort to provide a robust notation layer to the Quintet.net and has first been used in 2007 in a performance at the Budapest Kunsthalle. MaxScore went through several iterations and incarnations (e.g. as LiveScore bringing standard music notation to the Ableton Live DAW). For clarity’s sake we will now use the following nomenclature: *MaxScore* denotes the environment consisting of numerous objects, abstractions and scripts while the name *MaxScore object* refers to the Max Java object called com.algomusic.max.MaxScore. The MaxScore object is based on JSML, a language developed by Nick Didkovsky. I contrast to other Max notation solutions it requires a canvas to which to draw to and receive mousing information from. This “division of labor” affords greater flexibility as it allows the MaxScore object to render to various targets, such as Max drawing objects (lcd, jsui, jit.mgraphics) as well as SVG and PNG files, the latter via Jitter matrix export). Drawing commands, specific to the environment they are being executed in, can be defined as *rendered messages* and attached to notes, staves or measures.

In a 2018 paper, Hajdu and Didkovsky describe how scores generated by MaxScore could be displayed in real-time on iPads and browsers via the Max Mira/MiraWeb systems (footnote: reference to Carey and Hajdu: NetCanvas). This approach relies on the Max fpic object which can be mirrored on handheld devices. Having to create a PNG of the entire score each time it changes and using Mira’s and MiraWeb’s TCP and/or web socket connections for upload seemed like crutch and mandated a more elegant approach harnessing the power of modern browser with their JavaScript/HTML 5/SVG implementations. Our efforts thus led to the development of hfmt.drawsocket, which is a robust node.js-based solution allowing on-time delivery of scores that can be scaled and animated without loss of quality (due to the use of vector graphics).

Symbolist (related to SVG/OSC work) [2, 3] node.js prototypes used for ensemble mosaik

3. DRAWSOCKET

Development considerations  
Decided to use the browser, explain why?  
native server/client browser system leverages the massive

amount of development already done for web interaction o.io wrapper for the browser[4, 5]

6. MAXSCORE

As mentioned in 2., MaxScore possesses a fair amount of flexibility in terms of rendering to a wide array of targets. The JavaScript object *render2Browser.js* was created to facilitate the communication between the MaxScore object and the hfmt.drawsocket abstraction. The js object was designed with MNMP in mind. Such performances pose enormous difficulties when distributing large scores with dozens of staves. In performances with Quintet.net, scores containing just a few staves were split into instructions to be reassembled by individual instances of the MaxScore object and rendered locally by the Clients. But doing the same with dozen of instances (potentially destabilizing the environment and introducing unwanted latency), we resorted to a different strategy by implementing the concept of *multi-client rendering*, treating the ensemble of clients like one single canvas. In Maxscore, nearly every rendering message contains indexes referring to the notation object it represents (Example 1). Thanks to those indexes, *render2Browser.js* is capable of dynamically reroute a rendering message to targets set by the *staffgroups* attribute. This attribute can have the following values: *score*, *parts* or a list containing either indexes (for individual staves), two indexes joined by hyphens (for a staff range) or any number of indexes joined by plusses (for arbitrary collections of staves) such as in this example: staffgroups 0 1-2 2 0+3.

In addition to splitting and routing messages, the object respaces its staves so that resulting layout looks acceptable. It does so by querying the MaxScore object during rendering to obtain crucial information about staff spacing and using this information to apply offsets to the y values of each message to be rendered.

tempoqtrequals 20. 21. 0.5 Measure 0.

…

tr 22. 75.959999 0.5 Staff 0. 0.

staffnumber1 0. 63. 0.5 Staff 0. 0.

timesig4 43. 57. 0.5 Staff 0. 0.

…

StaffLine 0. 0. 4. 0.5 20. 75. 300.660797 75. false

…

frgb 0 0 0

noteheadblack 83.620689 57. 0.5 Note 0. 0. 0. 0.

frgb 0 0 0

no\_accidental 75.555557 57. 0.5 Note 0. 0. 0. 0.

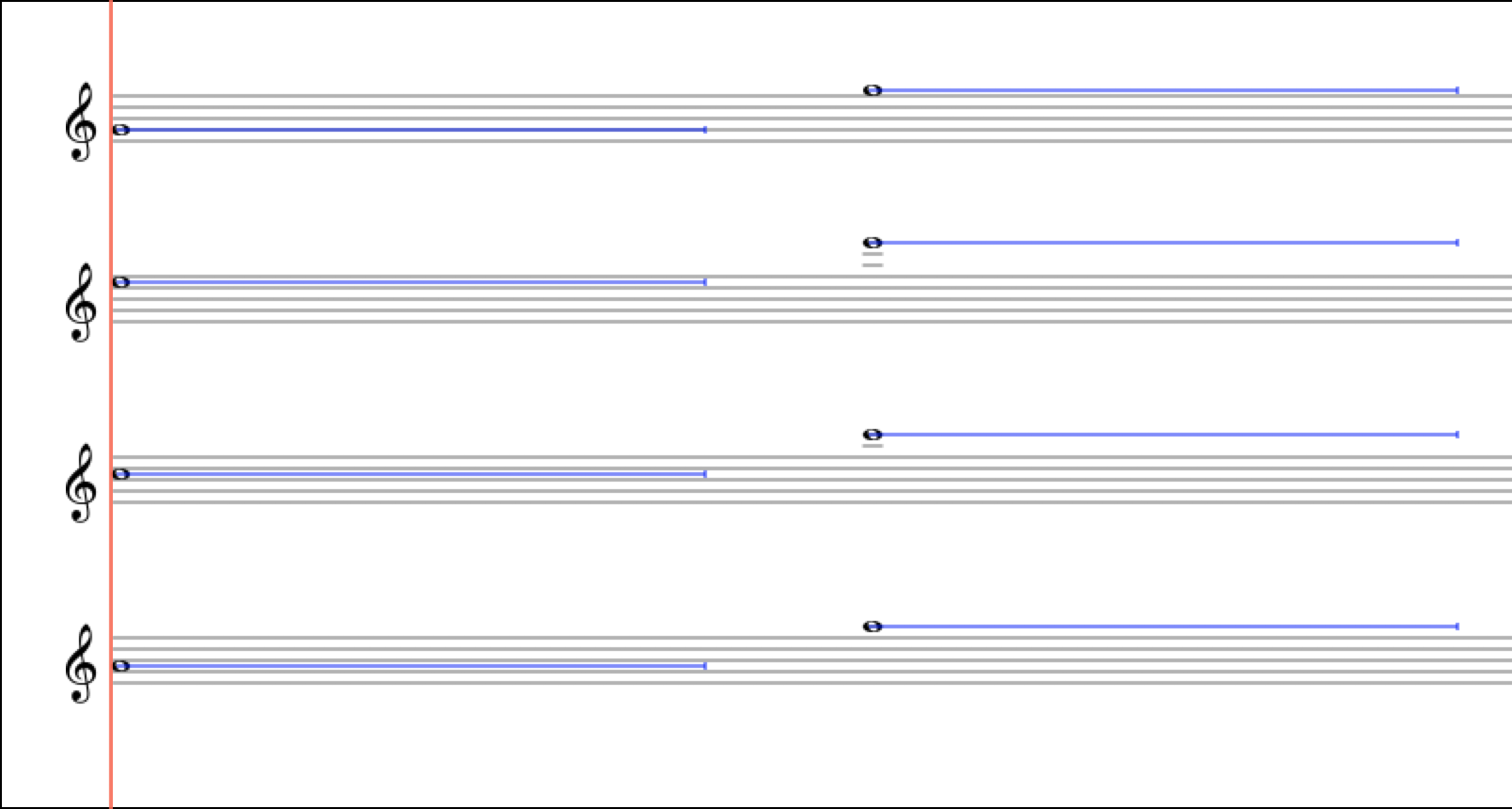
frgb 0 0 0

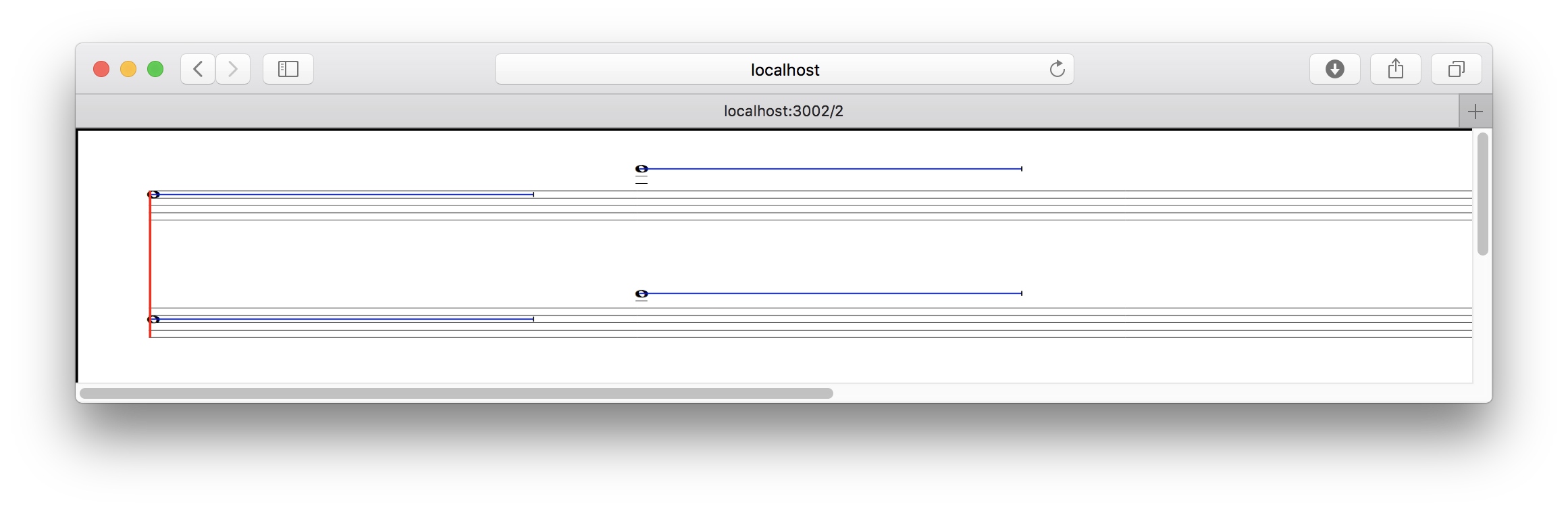
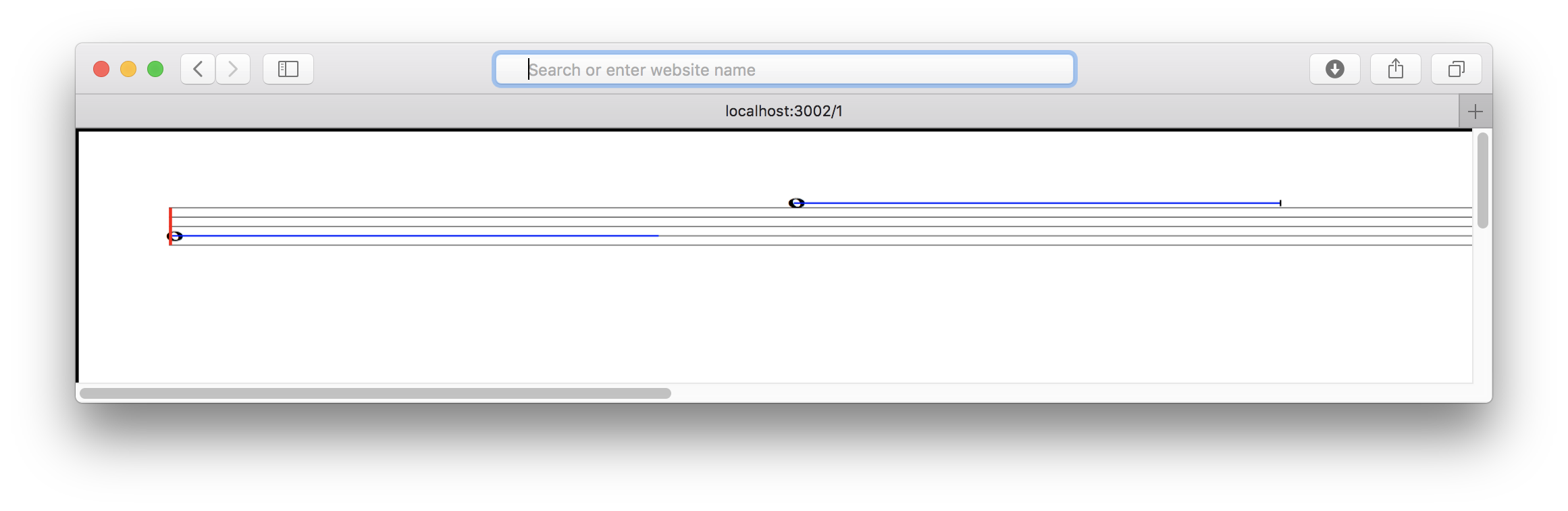
stem 76.620689 79. 0.5 Note 0. 0. 0. 0. STEM\_DOWN

RenderMessage staff 0 0 166. 13. 0.5 rendered Picster-Element[5] 175.3ocUOsnBBCCCzOk6GPwpap3Oyn1kMKtlHoE7E9uaaUTj.I4tK2A4r2ESjNmln.wIrGOvfZCjRtT2xJ1Igfk6iUPg5LwcQ+cBqfICcxjnYgRYxb6QwrkGmHX1hU6v5FztAqyJwjJmH7DddPpAJpezy40hcadZparTS6PUv7i3numpbI0xwAQCcA4CUuZuzYuRw+O3alu6OgkcGE82+roAaMKZQCtk8VtHWydAg7xSGC

Example 1. A sample of rendering messages generated by the MaxScore object. Note that nearly every message is accompanied by indexes (in red) referring to the notation object they represent. The y coordinates (blue numbers) are remapped according to the current staffgroups setting. The RenderMessage message contains a gzip’ed JSON object which in turn codes for a graphical score element such as a line, a rectangle, an arc or an image.

*render2Browser.js* is also capable of animating any number of cursors moving across set of measures and staves (see TENOR 2018 paper).





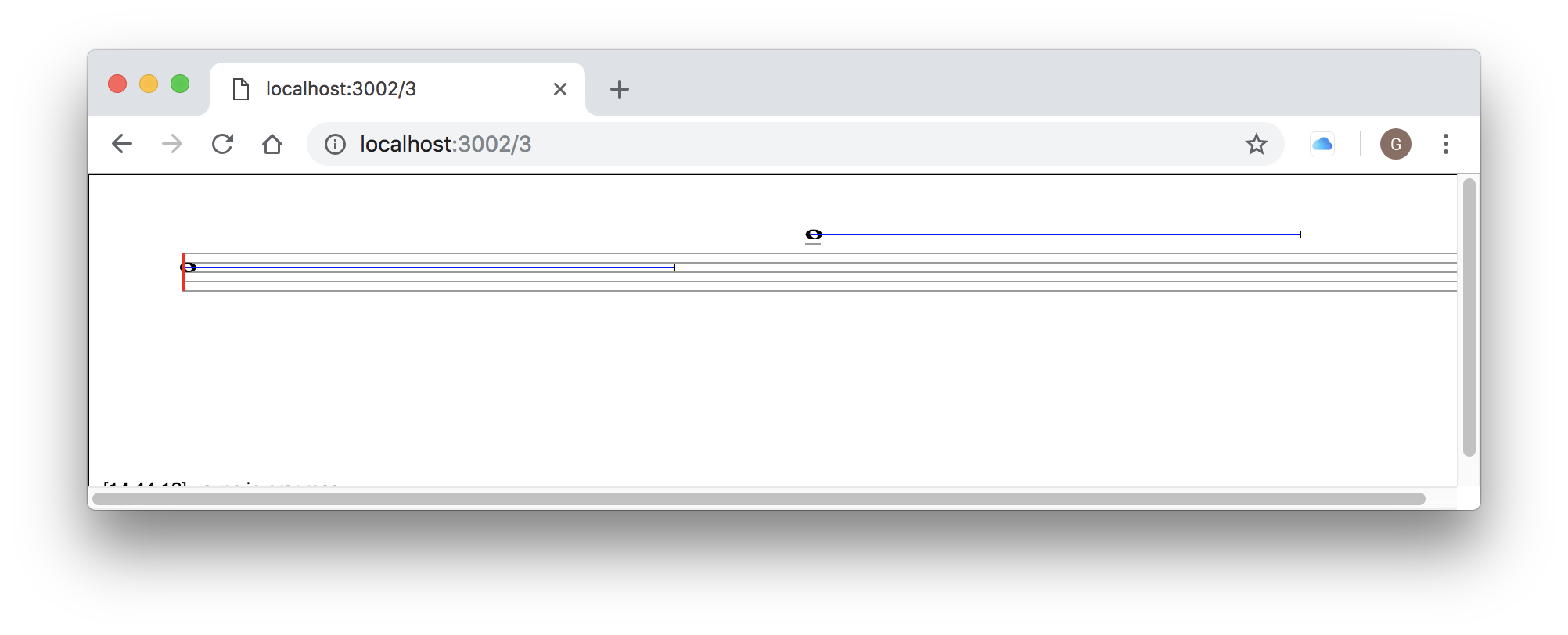
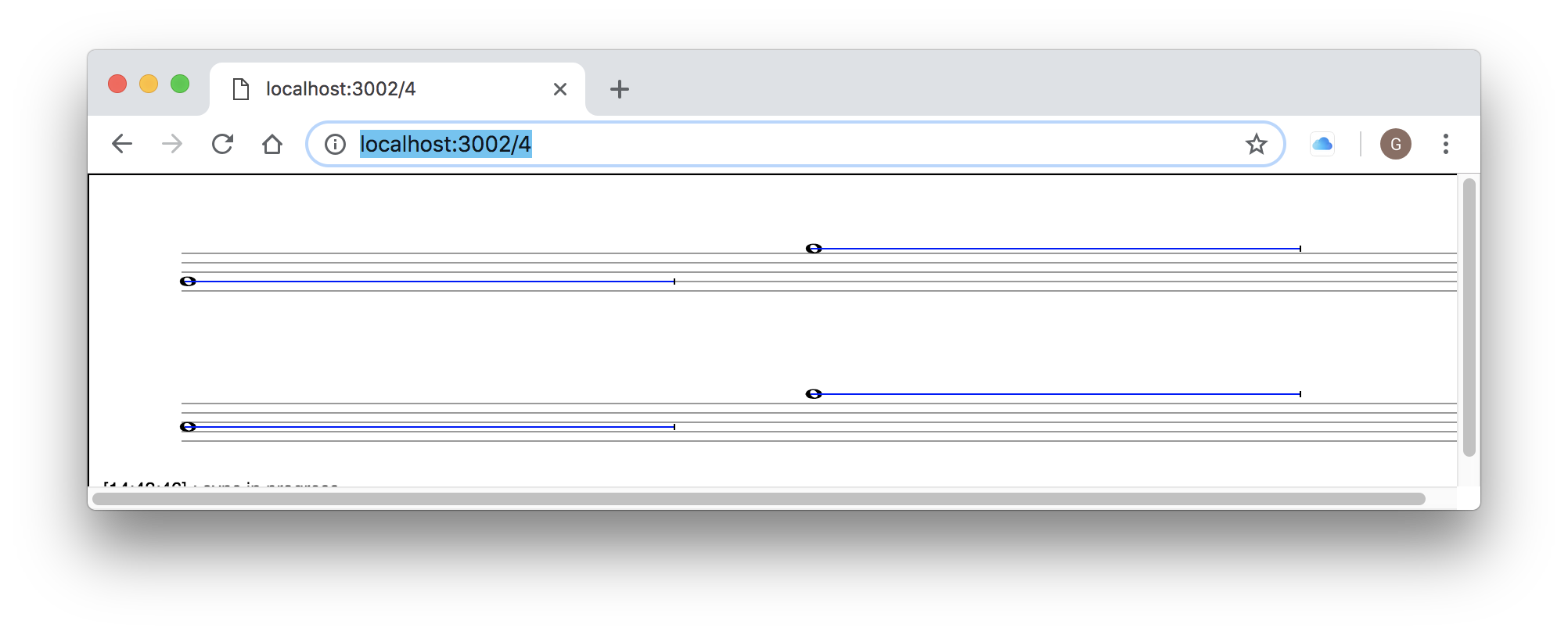
 

Figure x. A MaxScore score with 4 staves (top) rendered dynamically in four browser windows (center and bottom). The staves are split and grouped by render2Browser.js according to the following staffgroup settings: “0 1-2 2 0+3”.

To scroll the entire score horizontally, we created another JavaScript object called *maxscore.proportionalNotation*.js. It toggles between MaxScore’s default score layout and its proportional representation by hiding rests, stems, beams and naturals and indicating the duration of a note by a line extending from a note. The length of a measure is calculated by obtaining its tempo and time signature values and taking a *setTimeUnit* attribute into consideration. The durational scaling base value of 0.385 has proven to be optimal for spatially representing the delta time between events. The *start* message will cause a playhead to appear at the position given by the *scoreLeftMargin* attribute and instruct the browser to scroll the score. We are planning to also support scores created for the Decibel ScorePlayer in the future.



Figure 2. A score with a random melody rendered in MaxScore’s default layout.

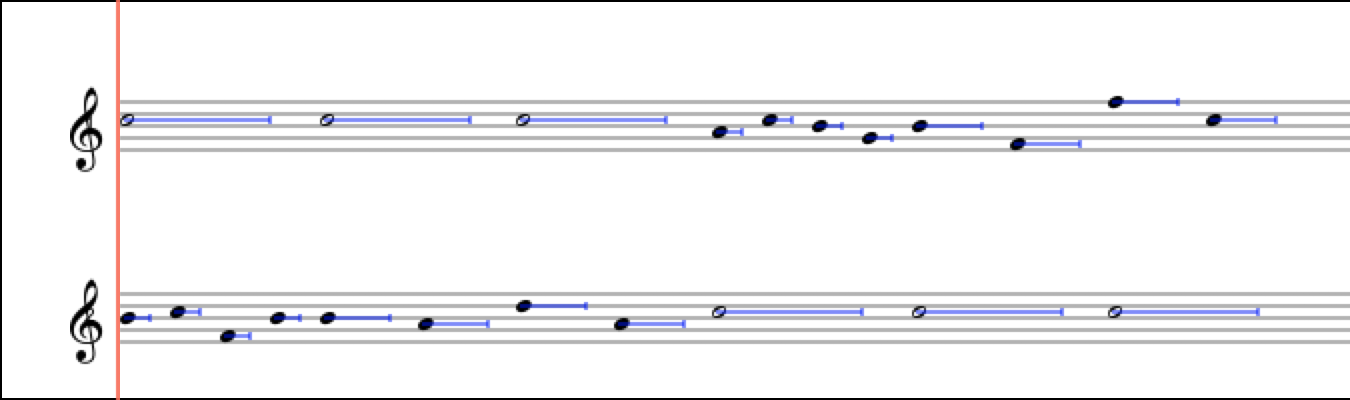


Figure 3. Same score after applying proportional notation. The default hold times indicated by the blue line is set to 80% of the event’s nominal duration.

7. CASE STUDY

A new multimedia piece (*Raindrops Keep Falling*) by Georg Hajdu was premiered at the December 2018 WOCMAT conference in Hsinchu, Taiwan. This composition for clarinet, cello and percussion consists of a transition between various rain samples and a late-1960’s hit called *Raindrops Keep Fallin' on My Head* mediated by a Max Pluggo effect called *Raindrops*. It’s also a tongue-in-cheek reference to the usual end-of-year weather pattern in Taiwan. The piece features 12 different versions of the song found on the Internet. The HfMT graduate student and research assistant James Cheung arranged the songs in such manner that they all share the same tempo structure and key signature, allowing the seamless navigation between those versions. James also created an arrangement of the song for the aforementioned instrumentation to be performed simultaneously with the recording, which was further subject to processing. First, parts of the score were “whited out” by a probabilistic process so that more and more events were allowed to appear paralleling a similar process applied to the audio tracks. The whiting-out was achieved by a JavaScript object called *maxscore.whiteout.js* capable of applying a “whiteout” gradient to a given section (the name was inspired by Cat Hope’s piece *The Great White*). Second, the score was turned into proportional notation, transmitted to the iPads of the performers via hfmt.drawsocket and scrolled in synch with the audio. The system held up to its promise as a computer-based conducting system. The scrolling was fluid and the musicians stayed in tempo despite the tempo fluctuations in the audio track.

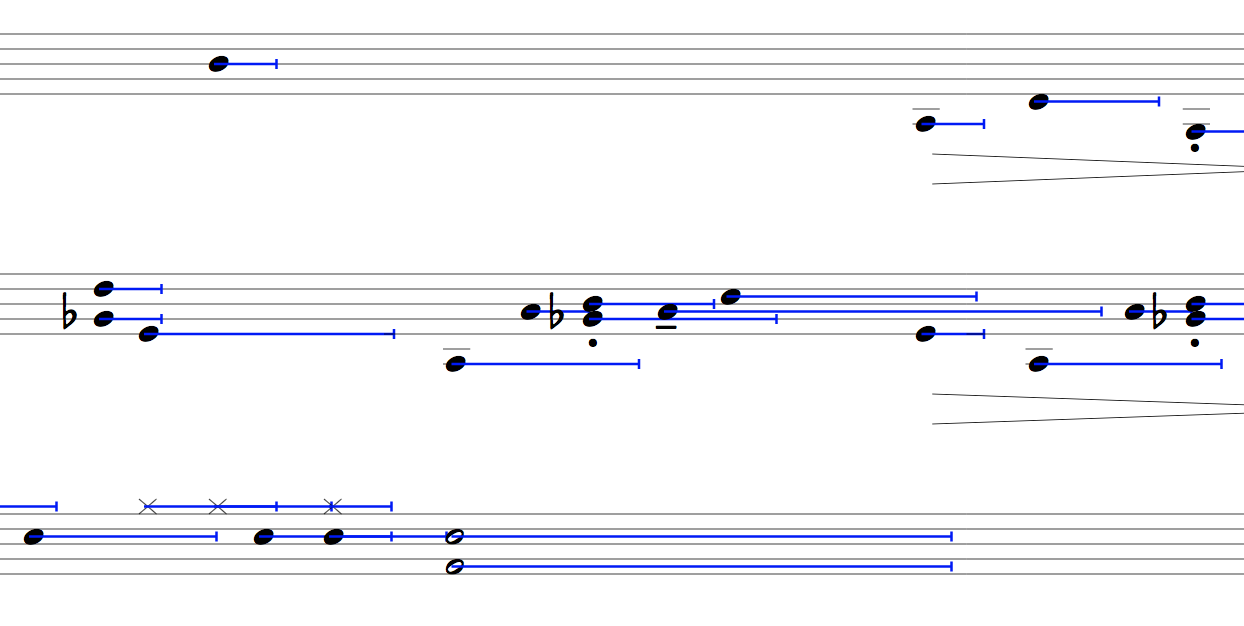


Figure x. Excerpt from *Raindrops Keep Falling* (2018) for clarinet, cello and drum set.

8. CONCLUSION AND PERSPECTIVES

[More testing needs to be done for large scale distributions, to determine how robust the messaging system is with many clients.

Mention WIFI research for St. Pauli tunnel here? ]

considering remove d3.js to reduce project size, since we’re only using it for DOM manipulation and basic attr/node creation, which could easily be done with vanilla JS.

considering changing API to focus on more complete object array system (nodes : [], and cssnodes : [] ) since this would be closer to the native SVG/HTML/CSS/JS code, giving many more options for the user. Also, using ex- plicitly named attributes rather than shorthand lists would make it easier for people to learn how to use the system by using the SVG specification reference directly. Note that this can almost already be done by setting a different base HTML page with a different default CSS form.

also consider adding an inject HTML option as well, a possible issue though is that we’d still need to select the newly created DOM object to store it in the node cache for fast access by GSAP and/or other manipulations.

at the moment the computation is offloaded to the client