NETSCORE: AN IMAGE SERVER/CLIENT PACKAGE FOR TRANSMITTING NOTATED MUSIC TO BROWSER AND VIRTUAL REALITY INTERFACES

Benedict Carey
Hochschule für Musik und Theater Hamburg
benedict_carey@icloud.com

Georg Hajdu
Hochschule für Musik und Theater Hamburg
georghajdu@mac.com

ABSTRACT

NetScore is an extension of the existing *MaxScore* package (Hajdu, Didkovsky) which adds new functionality for the rapid transmission and display of music notation on remote devices through standard modern browsers with WebSocket support. This was seen as a necessary development for *MaxScore* due to the ubiquity of tablets and other mobile devices, among other advantages for the user, and future applications of the software. We chose a server based solution executed in *Java* using the *Jetty* library for both portability between different platforms, and scalability. Novel applications facilitated by NetScore include transmitting scores generated in Max/MSP into virtual reality interfaces and more convenient performance/ rehearsal of real-time generated music, whereby devices commonly on hand such as smartphones, tablets and laptops are used as e-scores without requiring the installation of additional software.

1. INTRODUCTION

At the time of the emergence of Real-time composition as a trend in the 1980s, proponents of the approach had to have extensive knowledge of computer systems, or have access to a technician that could assist them realise their compositional ideas. Of the systems that did not require knowledge of command line scripting and other advanced techniques, few of them were designed for notation generation and the concept of real-time composition was to be a difficult to realise goal until much later with the advent of *OpenMusic* and *PWGL* [2]. In contrast, with recent developments such as the *Bach* library of Max/MSP/Jitter objects [1], *MaxScore* [2] and *Abjad* [3] and interfaces that can work with web browsers to display score data such as *INScore* [4] and *Scribe JS* (https://cruncher.ch/blog/scribe/), real-time musicians have experienced a veritable explosion of devices with which to perform from and produce complex symbolic notation in real-time. These tools are certainly far more accessible today to the average computer literate composer than those that came before them. Yet this technique remains in part highly technical, and while not prohibitively so, the associated configuration, installation and customisation of new compositions in addition to reconfiguring those compositions for ever changing platforms consumes much time for the composer. It may be the downfall of rehearsals and in worst case scenarios, actual concert performances should performers feel uncomfortable with the technologies they are asked to use.

A desirable outcome of the NetScore project was to create a score viewer that required no advanced knowledge on behalf of the person expected to operate and read the score. When considering developing a solution for MaxScore that supports mobile devices, we therefore considered it essential to develop a solution that did not require any extra installation or configuration on the users part, was platform independent and did not consume significant amounts of system resources. MaxScore, long associated with one of the authors' other similar projects *Quintet.net* [6], is at the heart of *NetScore*. MaxScore is a Max object, which accepts messages that can create a musical score, add notes to it, transform the notes, perform, save, and load the score, and export the score to popular formats for professional publishable results (for further info see www.computermusicnotation.com). In addition we chose a browser-based solution due to the ubiquity of mobile devices such as tablets and mobile phones, since many mobile devices available today contain a web browser of some sort. According to gsmaintelligence.com at the time of writing this paper there were as many as 3,763,381,520 unique mobile subscribers using a browser of some kind to interact with web content. It is fair to say then that these devices have indeed become ubiquitous; this number will no doubt continue to grow. The advantage of using mobile devices to display real-time notation is not just to do with how wide spread these devices have become.

The WebSocket Protocol (Fette and Melnikov, 2011) allows two-way communication in browser-based applications across a single HTTP connection. This protocol allows for the smooth interactions between server and client required for musical performance and reduces network overhead when compared with prior methods such as polling. According to StatCounter the most used browsers between October 2014 and October 2015 were Chrome, Safari, Internet Explorer and Firefox respectively (Figure 1), interestingly all of these browsers offer support for WebSockets making this choice of protocol a relatively safe one in terms of future developments of NetScore.

The music community is not without existing solutions to the problem of interacting with real-time notation through browser interfaces however. *Flight*, *Melodius*, *Scorio and Flat*, are all commercial solutions that offer in browser score viewing functionality. The GUIDO HTTP server exposes much of the GUIDO API

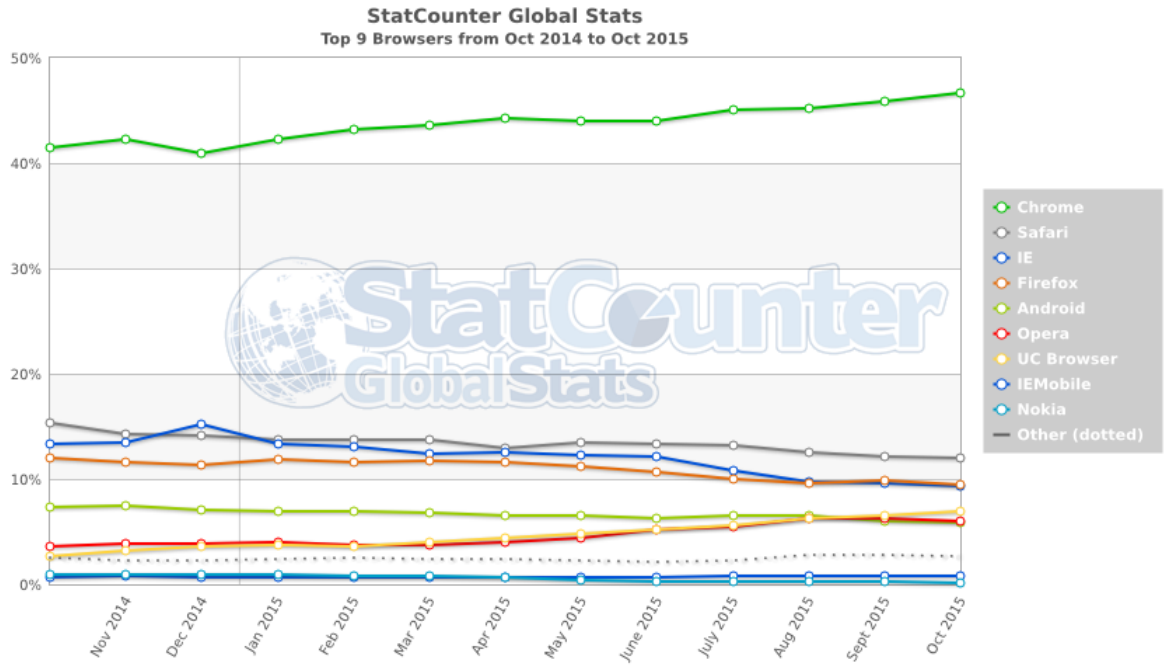


Figure 1. Browser usage statistics between October 2014 and October 2015, these popular browsers all support WebSockets a useful development for composers of real-time generated scores.

making it possible to create scores online using only open source software [7]. *Scribe* (https://cruncher.ch/blog/scribe/) a JS Library that allows the user to render music notation directly within web pages. *LeadSheet JS* performs a similar function but leans towards Jazz Leadsheet production [8]. INscore now offers WebSocket support and even in-browser interaction directly with the score material itself [9]. These solutions are highly customisable but they remain specific to the software packages that they are a part of and are therefore restricted to their own intended scope. The MaxScore package was already designed to work over networks due to its close relationship to the development of Quintet.net. In a technical sense, its separation of the graphical and symbolic representations of scores, meaning that the score files can be efficiently rendered to disk as PNG files, is very useful for network music applications. As the scores can be quickly rendered as graphics files directly from the LCD object in Max/MSP, adding this functionality to the MaxScore package does not increase processor overheads significantly on the host machine.

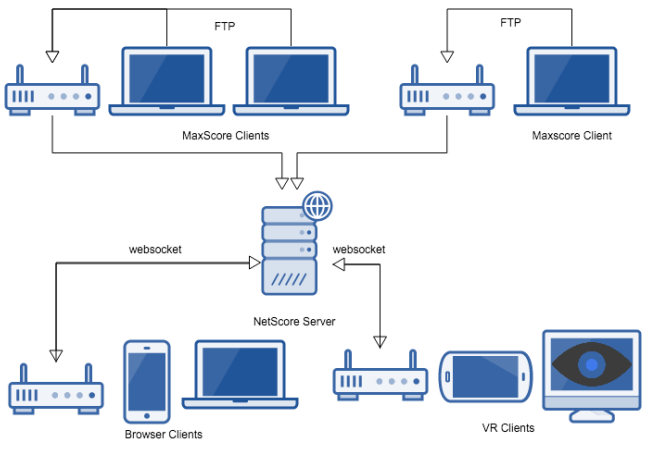


Figure 2. An example setup with NetScore, making use of its modular architecture to enable multiple users to interact remotely across a range of software and hardware platforms with minimal setup

2. NETSCORE IMPLEMENTATION

2.1 Four Separate Modules

NetScore operates using four separate components: a MaxScore Client, NetScore Server, MaxScore NetCanvas, and MaxScore VR-Client. Communication between these components is facilitated via FTP and WebSocket connections (Figure 2). With scalability a key focus of this project already, NetScore is based around a similar modular architecture, which allows for a high level of customisability. This lets users to explore a wide variety of composition and performance scenarios found in real-time and non-real-time composition such as live solo or group performance with e-scores, network music rehearsal and performance and VR collaboration. In fact a dedicated server terminal is not required, and one can even run all four components on the same workstation simultaneously if desired. This kind of setup is useful for testing a composition environment in its early development phase for example, as

one can easily transport the various score data to wherever it is required using the server.

As platform independence is an important consideration in Max/MSP, MaxScore and Quintet.net, the NetScore Server application itself is executed in Java, much like JMSL [10], which MaxScore relies on for data processing and storage. The NetScore helpfile is a typical Max/MSP helpfile. It provides an example of how to generate a MaxScore NetCanvas file though automated editing of HTML and JavaScript files, creating a small webpage. This webpage is customised with the websocket address of the host server. The resulting html file can be transferred to the potential client via whichever method they choose. Users are encouraged to create their own implementation, so as to take advantage of the flexibility offered within the Max/MSP and MaxScore environment. The VR client was created using Unity 5 and currently supports the Oculus Rift DK2 (https://www.oculus.com/en-us/) with front mounted or tabletop mounted Leap Motion camera on OSX, and Google Cardboard on Android, with a view to create a Windows and iOS version down the track. It relies on the C# WebSocket implementation *WebSocket Sharp*, which provides bi-directional communication over a single TCP Port as is needed for use with the NetScore server. (https://github.com/sta/WebSocket-sharp). The user must manually select the IP of the WebSocket server in order to achieve a connection.

2.2 WebSocket Image Server Implementation with Jetty 7

The advantage of running a separate application as our WebSocket server is that multiple remote connections are possible, allowing for scenarios where different MaxScore users are concurrently creating scores and uploading them to the appropriate locations on the server, which are being continuously scanned for changes. The user is still free to run the app locally of course, which could be useful as in a situation where local performance is to be the outcome and the networking functionality is less important. We explored the potential for using node.js from within Max, as well as other Java WebSocket libraries, which were supported with the older JVM. In the end we adapted source code developed by Desmond Shaw as the basis for the server application (http://www.codepool.biz/how-to-implement-a-java-WebSocket-server-for-image-transmission-with-jetty.html), which we then modified to suit our purposes adding functionality for handling the folder scanning operations and our own filename referencing system for requesting scores from the client side. After the handshake is completed the client transmits a part number request (user definable through the client interface) and the desired part is delivered. The server understands any integer as an instruction starting a part request, where a "0" stops the server from scanning the folder for changes for that particular client. Each WebSocket connection is handled in a separate worker thread. In the console window messages are displayed indicating the network activity of all connected clients, and notifies the user of any file changes. Only one copy of the server application may be running locally at any one time per machine.

2.3 Image Handling

The PNG files generated by MaxScore can be forwarded via FTP or locally copied to a specific folder hierarchy on the terminal running NetScore Server with a filename corresponding to which part they represent. The server in turn creates WebSocket connections to clients who have successfully completed the handshake and reports their IP to a console window, and sends the requested score whenever it changes. These PNG scores are "pushed" to the clients as byte arrays from the server, where they are converted into images, and are either applied as a texture in VR via a C# Unity script or displayed in a browser window. Graphic scores, with their larger file sizes are better supported by this method than if the page was to constantly refresh itself in the off chance that the score file on the server had changed.

Currently PNG and JPEG formats are supported by the Server application, and the user can choose to manually transmit these file types to all connected clients using the *Load* and *Send* buttons in the server GUI, for testing purposes (Figure 3). Images are translated into byte arrays, transmitted and then converted back on the other end by the client into graphics as with other connections. A separate thread is run per connected NetCanvas Client or VR client instance instead of it restricting an IP address to a single connection, meaning that a client can open multiple browser windows and simultaneously stream different parts. On the client side, in the MaxScore NetCanvas for example, the byte arrays are reloaded into the page as images.

2.4 Client and Server Graphical User Interfaces

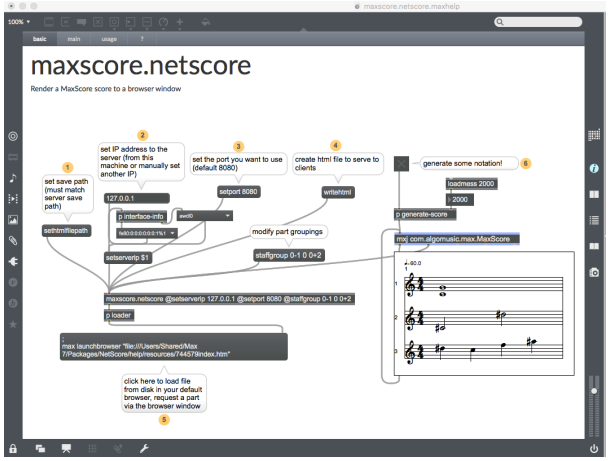


Figure 3. Running multiple browser windows displaying different parts.

2.4.1 Max/MSP Help File

The Max/MSP help file illustrates the execution of a real-time composition using MaxScore and NetScore to customise a website to send to clients, and generate a composition. The user can follow the steps outlined to obtain the current external or local network IP address, which can be used to build a small webpage that is sent to users via email or other methods such as Apple's *AirDrop* or local file transfer. The file path where the score files are to be saved must be specified here, but alternatively the user can opt to send their image files to the MaxScore Server via FTP. An example is given whereby the user can generate a random real-time piece of music with the help file.

2.4.2 Server Side

The MaxScore Server has a simple interface consisting of a load and send button mainly for testing if a connection has been achieved. On launch, the user is prompted to select a location for the score files. This is the folder that will be scanned for changes in files with the appropriate file names and extensions. The main window displays messages regarding the connection status of clients and folder activity. It also reports every time it forwards an image to a client and when a text message or file request is received from a client. Any errors associated with the connection are also reported here.

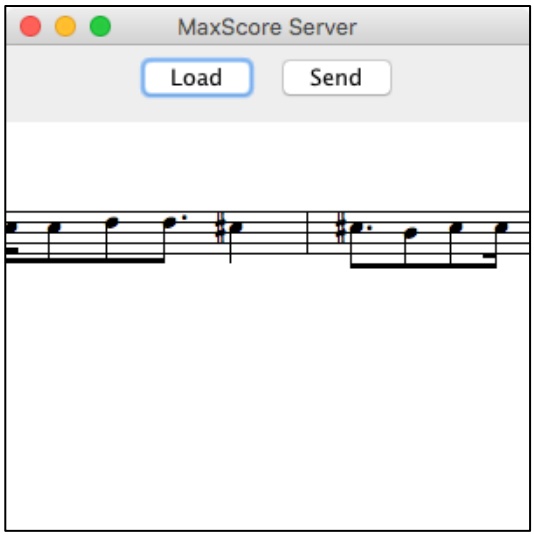


Figure 3. The Server interface, with its manual load and send buttons, displaying a score file about to be sent across a network.

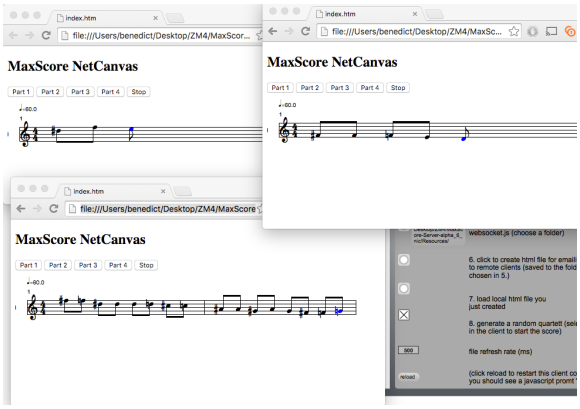


Figure 4. Running multiple browser windows displaying different parts.

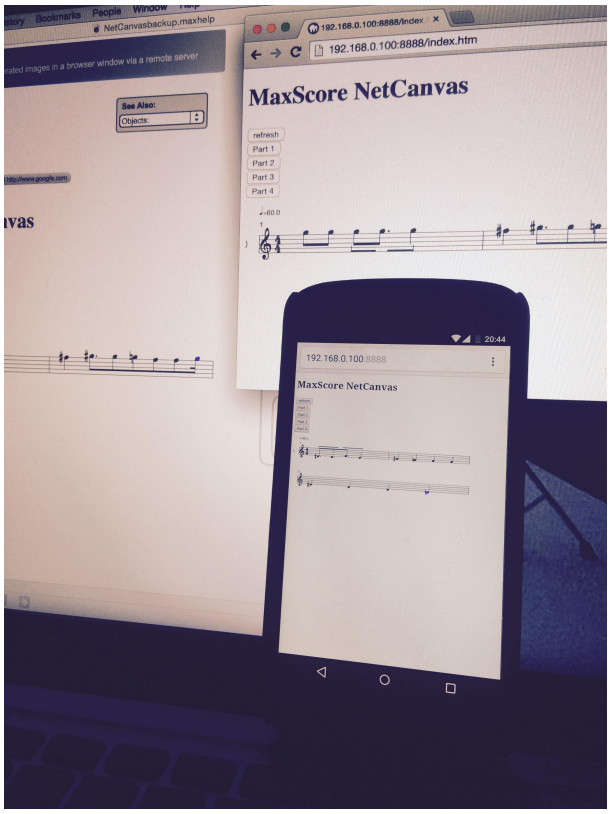


Figure 5. MaxScore NetCanvas. This browser based image client makes it possible to view realtime generated scores (or any picture file for that matter) on iPads, iPhones, Android mobiles, Laptops - basically anything that can run a modern browser.

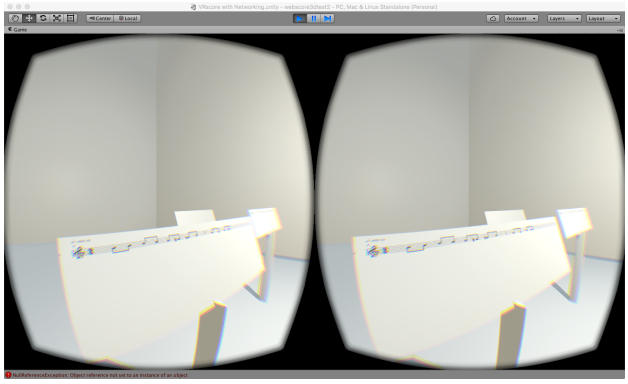
2.4.3 MaxScore NetCanvas

The MaxScore NetCanvas (Figure 5) consists of a number of buttons for requesting scores from the server, a score title and the contents of the most recently delivered PNG file. Since it is created based on a simple HTML and JS webpage, the user is free to customise this interface through text editing of the original resources, located in the installation folder. Currently the

demo interface supports selection of separate parts and a stop refresh button (figure 4). These can easily be pinched to enlarge or flipped to rotate the orientation on iOS and Android devices as is typical of a normal webpage, if it is supported by the device.

2.4.4 Virtual Reality Client

The Virtual Reality client is built in Unity 5 as a collaboartive environment reminiscent of the stage layout of past Quintet.net performances (Figure 5). Although the MaxScore Server is required to serve scores to the environment, (in the same fashion as scores are served to browser clients), the actual multiplayer functionality is handled by Photon server (https://www.photonengine.com/en/PUN). This free service supports up to 20 simultaneous player connections and delivers a high performance low latency interaction. Upon launching the application, each player spawns into the next available spawn spot, set up in front of a virtual score. It is currently possible to track the users hands if they are using a leap motion controller to facilitate conducting or performance of virtual reality instruments designed in Max/MSP. Each VR score in the environment has an attached C# script which handles both the websocket connection and applies the incoming byte array as a texture to the object to which it is attached (Figure 5). This means that for each connected player, the scores in their own local version of the client will change in time, even if latency has affected the synchronization of a fellow player's avatar. A Leap motion controller can be connected in the case of the OS X client, whose movements are mapped directly onto the avatars skeleton, creating convincing interactions appropriate for a convincing VR performance. Players are able to select virtual instruments running in Max/MSP, communication is facilitated with a customised version of the *Max/Unity Interoperability Toolkit* [10].



```
using UnityEngine;
using System.Collections;
using System;
using WebSocketSharp;

public class socketTexture : MonoBehaviour {
    // Use this for initialization

    public int partNumber;

    IEnumerator Start () {
        WebSocket toJava = new WebSocket(new Uri("ws://192.168.0.10
        yield return StartCoroutine(toJava.Connect());
        int i=0;
        Debug.Log ("Connected");
        yield return new WaitForSeconds(partNumber);
    while (true)
    {
        if (i<100) {
            {
            String fromSt = ("0");
            toJava.SendString (fromSt);
            //Debug.Log ("Sent:" + partNumber);
            yield return new WaitForSeconds(1);
            string toTex = toJava.RecvString();
            Debug.Log ("Received: "+ toTex);
            var tex = new Texture2D(2, 2);
            byte [] outTex = Convert.FromBase64String(toTex);
            tex.LoadImage(outTex);
            // Assign texture to renderer's material.
            GetComponent<Renderer>().material.mainTexture = tex;
            yield return 0;
        }
        }
        if (toJava.Error != null)
        {
            Debug.LogError ("Error: "+toJava.Error);
            break;
        }
    }
    toJava.Close ();
    }
}
```

Figure 5. – Above, C# script to receive texture data via WebSocket and apply to surface, below, Virtual reality client with notation texture applied to music stand.

3. FUTURE DIRECTIONS

Although the NetScore package itself offers some new possibilities for MaxScore users, it is still in its earliest phases of development. Firstly, there is a great deal of functionality we could still implement. Interactive elements from within the browser based canvas as well as the VR interface could take the project even further in the direction of the Quintet.net package. Even during the development phase we have already seen changes to how iOS handles HTML files via email for security reasons and have had to alter the way the customised MaxScore NetCanvas files are served to them. Keeping abreast with developments across multiple platforms presents additional challenges that will hopefully be made easier through our deliberate reliance on Java and Unity, which facilitate fairly rapid development across platforms. The VR client also has potential in the future to be used as a basis for further experiments and artistic work, and could hopefully become better integrated with the existing *Quintet.net* software, where it could act as an input device in addition to handling score display.

On the other hand using *libwebsockets* (https://libwebsockets.org/index.html) it may be possible to create a MaxObject that acts as a WebSocket Server but so far no implementation has been attempted. Another version of the server relying on MassMobile may be another way to achieve this [12]. If such a solution were possible, the NetScore package could be significantly simplified, without a loss of functionality.

It remains to be seen how NetScore will perform in performance scenarios, and benchmarking is planned for the future. The primary purpose of this proposed research would be to reduce latency to the point where scores can be delivered at a much higher rate than is possible with the current system, which utilises a 500 ms refresh rate, far to slow to facilitate animation for example, but certainly suffient for the puposes of extreme sight reading [11]. It is without a doubt a very exciting prospect to be able to control Ableton Live in LiveScore, a package of Live devices built with MaxScore, with our VR interface, so as to free the real-time composer from the distraction of a laptop screen, mouse and keyboard interaction system. Additional efforts to embed the fonts used for symbolic representation within MaxScore in SVG files using the Apache Batik SVG toolkit (https://xmlgraphics.apache.org/batik/) could potentially introduce client-side control over the resolution of scores. Using the ol.wsserver (http://olilarkin.blogspot.de/2014/01/olwsserver.html) object from Oliver Larkin or similar could handle resolution or other rendering specific requests to Max/MSP using the current system however and would be somewhat easier to implement.

4. CONCLUSION

By extending MaxScore to mobile devices and a virtual reality interface, the authors have achieved a welcome upgrade to an already feature rich software package with over a decade of previous development behind it. The ability to integrate notation into virtual reality environments is a novel development, and opens up a plethora of creative and technical possibilities to the Max/MSP/Jitter and Ableton Live communities.

Acknowledgments

The authors would like to thank he city of Hamburg for its financial assistance via its Landesforschungsförderung program and Desmond Shaw from Dynamsoft for contributing his source code.

5. REFERENCES

[1] A. Agostini and D. Ghisi, "A Max Library for Musical Notation and Computer-Aided Composition," *Computer Music Journal,* 2015.

[2] N. Didkovsky and G. Hajdu, "Maxscore: Music notation in max/msp," in *Proceedings of the International Computer Music Conference*, 2008, pp. 483-486.

[3] J. Freeman and A. Colella, "Tools for real-time music notation," *Contemporary Music Review,* vol. 29, pp. 101-113, 2010.

[4] D. Fober, G. Gouilloux, Y. Orlarey, and S. Letz, "Distributing Music Scores to Mobile Platforms and to the Internet using INScore," 2015.

[5] J. Freeman and A. Colella, "Tools for real-time music notation," *Contemporary Music Review,* vol. 29, pp. 101-113, 2010.

[6] G. Hajdu, "Quintet. net: An environment for composing and performing music on the Internet," *Leonardo,* vol. 38, pp. 23-30, 2005.

[7] K. Renz and H. H. Hoos, "A WEB-based Approach to Music Notation using GUIDO," in *Proc. of*, 1998, p. 98.

[8] D. Martín, T. Neullas, and F. Pachet, "LEADSHEETJS: A JAVASCRIPT LIBRARY FOR ONLINE LEAD SHEET EDITING." (https://csl.sony.fr/downloads/papers/2015/dani-15a.pdf)

[9] N. Didkovsky and P. Burk, "Java Music Specification Language, an introduction and overview," in *Proceedings of the International Computer Music Conference*, 2001, pp. 123-126.

[10] I. I. Bukvic, *M Max-Unity3D Interoperability Toolkit*: Ann Arbor, MI: MPublishing, University of Michigan Library, 2009.

[11] 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, 2008.

[12] N. Weitzner, J. Freeman, S. Garrett, and Y. Chen, "massMobile–an audience participation framework," in *Proceedings of the New Interfaces For Musical Expression Conference*, 2012.