MMODM: Massively Multiplayer Online Drum Machine

Basheer Tome
Tangible Media Group
MIT Media Lab
75 Amherst St.
Cambridge, MA 02142
basheer@media.mit.edu

Donald Derek Haddad
Responsive Environments
MIT Media Lab
75 Amherst St.
Cambridge, MA 02142
ddh@mit.edu

Tod Machover
Opera of the Future
MIT Media Lab
75 Amherst St.
Cambridge, MA 02142
tod@media.mit.edu

Joseph A. Paradiso
Responsive Environments
MIT Media Lab
75 Amherst St.
Cambridge, MA 02142
joep@media.mit.edu

ABSTRACT

Twitter has provided a social platform for everyone to enter the previously exclusive world of the internet, enriching this online social tapestry with cultural diversity and enabling revolutions. We believe this same tool can be used to also change the world of music creation. Thus we present MMODM, an online drum machine based on the Twitter streaming API, using tweets from around the world to create and perform musical sequences together in real time.

Users anywhere can express 16-beat note sequences across 26 different instruments using plain text tweets on their favorite device, in real-time. Meanwhile, users on the site itself can use the graphical interface to locally DJ the rhythm, filters, and sequence blending.

By harnessing this duo of website and Twitter network, MMODM enables a whole new scale of synchronous musical collaboration between users locally, remotely, across a wide variety of computing devices, and across a variety of cultures.

Author Keywords

Collaborative, Synchronous, Twitter, Web, Music Interface

ACM Classification

H.5.3 [Information Interfaces and Presentation] Group and Organization Interfaces — Web-based interaction, H.5.3 [Information Interfaces and Presentation] Group and Organization Interfaces — Collaborative computing, H.5.3 [Information Interfaces and Presentation] Group and Organization Interfaces — Synchronous interaction

1. INTRODUCTION

Music and, increasingly, social networks are both incredibly powerful categories of tools for connecting people together and creating positive flow-like experiences. We strongly believe that a more intimate overlap and connection between the two can be instrumental in better connecting even more and diverse types of people.

NIME'15, May 31-June 3, 2015, Louisiana State Univ., Baton Rouge, LA.

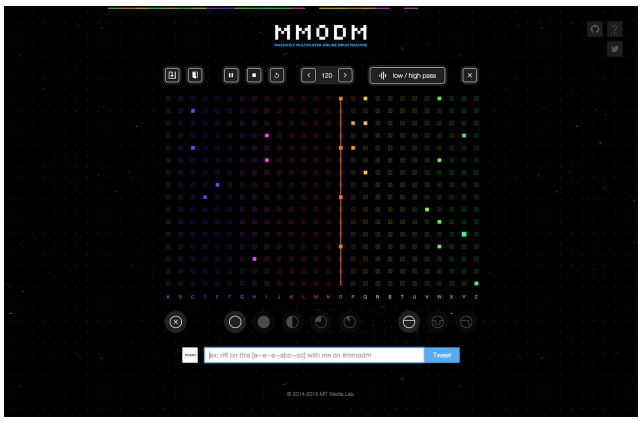

Figure 1: MMODM site during a live performance

1.1 The Rise of Twitter

Twitter usage has quickly skyrocketed since its invention back in 2006 by Jack Dorsey. Despite any negative uses of Twitter, it has also helped make a positive impact on society and has provided a variety of uses for members across the globe. During the earthquakes that took place in California, it proved to be one of the main sources of the most up to date activity of this monstrous natural disaster. There's no better way to share information than to have a large network of friends all connected in one system, who can reply to your messages and even retweet them to others at their convenience.

1.2 Related Work

The realm of technology-supported music collaboration is quite large but work in the spaces of live coding and web-based audience participation served as the most critical references in the creation of this project. The sc140 SuperCollider album [1], which curated a series of pieces written in tweet-length SuperCollider code snippets, shows at the extreme end the expressive ability of just 140 characters. Using that same programming language in a longer form, Magnusson [2] is able to live code complex, beautiful musical pieces in real time. Furthermore, the LOLC collaborative textual performance environment [3] shows the dynamic performative musical collaboration that expert music tech-

nologists can have together with powerful tools like these. We realize that the simplified language syntax of MMODM could never surpass the expressiveness and complexity of these programming languages but we seek to make it robust enough that the loss in complexity is worth the gains in accessibility to new musicians.

There have been other web-based collaborative music apps done both before and after Twitter's inception. One of the first was Eric Metois' collaborative sequencer, Spinning Disks, that he created for the MIT Media Lab's Brain Opera [4]. It allowed users to enable samples on rotating disks, visually, in an interactive client-side web applet using their mouse. Another example is Jorge Herrera's Horgie, a web-based Networked Music Performance tool that synthesizes multiple remote users' input on a centralized server through the Flash plugin [5]. On the mobile side, Nathan Weitzner's massMobile [6] enables audience members using mobile devices to draw the score collaboratively, in real time, for a live jazz ensemble. These examples are powerful tools, but, because they rely on a singular input & output method for each remote contributor controlling a synchronized musical space [7], they solve the distance limitations of the physical world but still exhibit some of the limitations in the scale and speed of participation. Twitter's ubiquity, latency, and diverse input device compatibility truly facilitate the ability to push collaboration to the next level. It's ability to flexibly work as a communication medium between communities of 10 to 10 million people is truly unique.

That said, works utilizing Twitter itself for music creation are also plentiful. A great example is The Listening Machine by Daniel Jones and Peter Gregson [8]. It is an automated system that generates a continuous piece of music based on the activity of 500 Twitter users around the United Kingdom. Their conversations, thoughts and feelings are translated into musical patterns in real time, which you can tune in to at any point through any web-connected device. However, while it and others served as solid inspiration for us during the creation of MMODM, they do not allow for real-time, intentional collaboration, and instead generate music incidentally from the interactions on the site. We propose that by giving users more direct, intentional, and active control through Twitter, we open up the possibilities for the type of music that can be created.

2. INTERACTION DESIGN

Because our multi-part system has asymmetric participation roles, we faced many issues and nuances in delegating tasks between site users and twitter users, in order to create a balanced dynamic between creator, editor, and audience. We outline below the user-facing features that flowered into our core set enabling sequence creation, closer collaboration & sharing, playback controls, filters & effects, and sequence modification.

Figure 2: Flowing color bar at the top of the page corresponding to new notes from the stream

The limitations of various browser rendering engines imposed visual and graphical constraints on what we were able to do visually. However, our small design details like the flowing color bar and pulsing colored squares representing the notes help to create the feeling of a more dynamic, collaborative experience using simple shapes and a simple color palette.

2.1 Sequence Creation

At the heart of MMODM is sequence creation. Twitter users anywhere can express 16-beat note sequences across 26 different instruments using plain text tweets on their favorite device, in real-time. To define a sequence, use square brackets [ ], a string of 16 letters A through Z corresponding to various pre-selected instruments, and add the hashtag #mmodm. There are 16 beats total in a loop and a "-" is used to define a rest. For example:

riff on this [a–a–a–abc–cc] with me on #mmodm

2.2 Sharing & Collaboration

Users can share the current state of their local session on the site through the share button in the top left corner. Clicking it locks the notes in place and generates a link (ex: mmodm.co/sm/ezwg8u5o). Sharing that link allows other users to both listen to the sequence as well as tweak it and build on top of it.

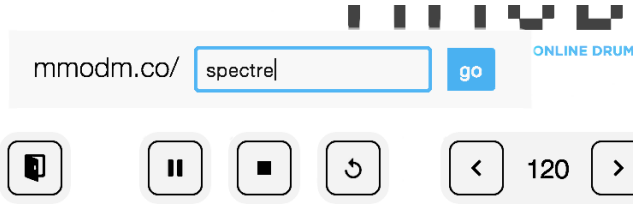

Figure 3: Interface popup used to create a hashtag-exclusive room to jam

When there are a lot of users using the #mmodm hashtag, it can be tougher to have a more intimate jam session with a few friends locally or remotely. By clicking on the rooms button (Figure 3), users can create a separate musical space requiring both the #mmodm hashtag as well as a hashtag of their choosing such as #spectre.

2.3 Playback & Tempo

Standard playback support of pause, play, and stop are included in order to aid in timing in performances and with other instruments.

Figure 4: Buttons used to control pause, play, looping, and tempo

Some interesting, looping effects can be achieved using the restart button (Figure 4) that restarts playback at the beginnign of the loop without interruption. Lastly, the tempo is also modifiable on a local level and can be combined with the effects (such as beat skipping) to create more interesting and complex sounds.

2.4 Filters & Effects

While many types of filters and effects were considered, a brief survey of what musicians actually used and what effects were most commonly found in other types of drum machines led us to incorporate a small but important subset of filters and effects [9]. We added high & low pass filters, stuttering, and beat skipping (Figure 5).

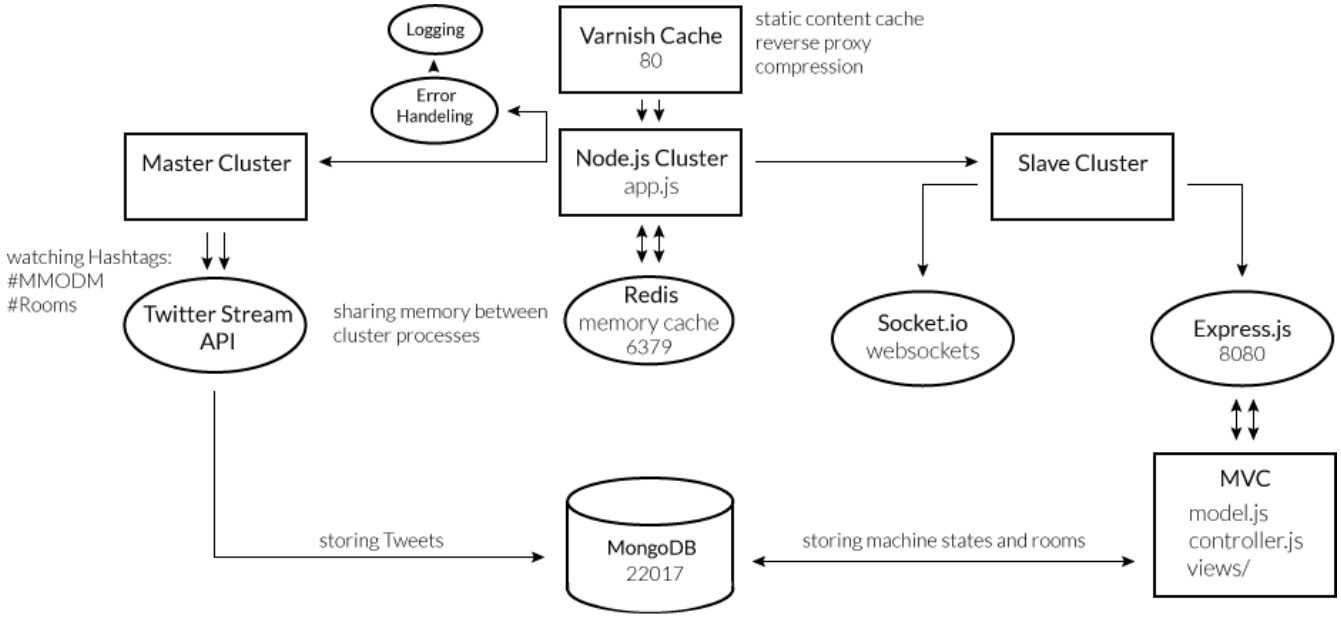

Figure 6: Flow diagram describing in detail the technical system architecture design

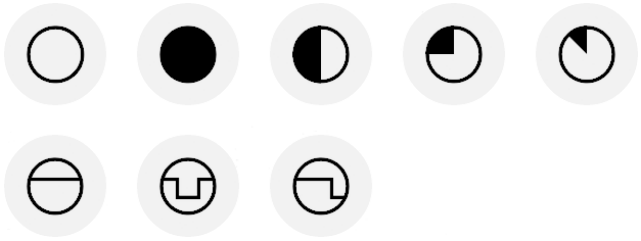

Figure 5: Buttons used to select between various ratios of stutter timing and beat skipping effects

The stuttering operates on a two-note buffer loop, with options to loop on-tempo, 2x, 4x, and 8x. The "gater" effect skips playing either every-other beat or every-two beats.

Furthermore, users can more expertly employ these effects by using the keyboard shortcuts (0-9) to speedily enable and layer.

2.5 Sequence Modification

Nearly as important to the sequence creation functions is modification through the column locking and sequence clearing features. (Figure 7)

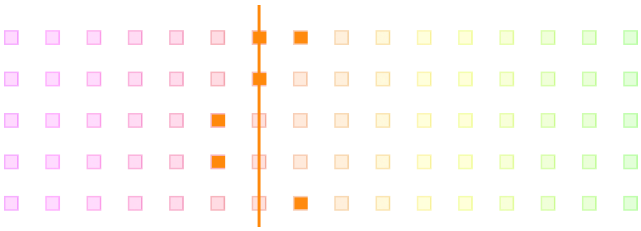

Figure 7: Clicking on a column locks those beats in place with a line, preventing them from fading out

By clicking various columns with their mouse, users can prevent changes to that note column and preserve instrument-specific sequences that are working well. This prevents the natural fading of notes over time, stops the clear button from removing those notes, and ignores newly tweeted notes of that specific instrument. It's through well-considered use of this feature that enables users to locally shape the chaos of collaboration into beautiful arrangements.

3. ARCHITECTURE

Unlike many other common web applications, MMODM relies almost as equally on the back-end server architecture as it does on the client-side javascript code running in the user's browser to create the full musical experience. This is necessary in order to avoid extraneous streaming connection requests to Twitter while also running the core timing loops locally to ensure speed, accuracy, and latency while jamming.

.

3.1 Server Architecture

The back-end architecture, as shown in Figure 6, relies on cutting-edge technologies to serve thousands of incoming requests. At its foundation lies Varnish-Cache, a web application accelerator, running on port 80 to reverse-proxy incoming requests to the Node.js server cluster. The Node server cluster runs the master and slave processes. The master cluster's main job is to keep slave processes up and running. It also, watches over the root hashtag (#mmodm) via Twitter's stream API, i.e., the master process will capture, process and store any tweet having this hashtag. The incoming tweets are checked by the master process using regular expressions to ensure that they fit MMODM's text-to-beat constraints outlined in the interface design section. If passed, these tweets are stored in a MongoDB database running on port 27017 before being broadcasted to connected peers via Websockets.

The slave processes spawn the core application layer running Express.js, a lightweight and modular Node.js web framework, and Socket.io enabling real-time bidirectional event-based communication. The Express.js http server runs on port 8080 for production and defaults to port 3000 for development or testing. Finally, on port 6379 Redis server, a key-value memory storage, unifies the memory pool used across the master and slaves processes enabling a horizontally scalable engine. The Express.js application layer follows a classical MVC design pattern [10] where Jade, an

HTML pre-processor and template engine, is used to organize views. Mongoose is then used to provide a simple object document model (ODM) for MongoDB [11]. Errors are handled by both master and slave processes: errors coming from Twitter's Stream API are handled and logged by the master process whereas app-routing errors are handled by slave processes.PM2 ,a production process manager for Node.js applications, is used to keep our web-app up and running by reloading it without downtime. It also facilitates common system admin tasks like logging and scaling.

The full server-side software stack includes:

- Varnish-Cache 4.0
- MongoDB 2.6.5
- Redis 2.8.19
- Node.js 0.10.36
- Cluster 0.10.36
- Express.js 3.3.6
- Socket.io 1.2.1
- Jade 1.9.1
- Mongoose 3.8.22
- PM2 0.10.12

3.2 Client-side Architecture

The client-side architecture is only compatible with modern browsers that support both Websockets and the web-audio API. The main Javascript file connects to our back-end server to update the browser's data object model locally as new Tweets flow across the drum machine as well as to pull the past few tweets from the cache. When incoming messages update the DOM, sound is generated by triggering each audio file that corresponds to the letter in that sequence as the interval-timed script loops through the elements. New notes decay after 30 seconds if not locked or newly tweeted as described in the interaction design section.

While sequences are distributed globally, timing and effects are local to a session in order to enable more controlled performance. Low and high pass filters are implemented on a per-note basis using web-audio's core filters functions.

Lastly, by using the live document itself as the primary local data structure, we open the opportunity for simple-to-implement user-based hacks and tweaks to the sound and functionality of the site through CSS stylesheets applied by browser extensions such as Greasemonkey or Google Chrome's drawer Console. For example: a holiday theme with a bright red background and bell notes.

4. CONCLUSION

Through a live and launched online application of MMODM at `http://mmodm.co/`, we have created a new tool for synchronous musical collaboration between users locally and remotely over Twitter.

By harnessing this powerful network, we're able to take advantage of its flexibility and simplicity. Our hope is that MMODM is able to flexibly scale from 5 users to 5 million both musically and technically, across a wide variety of computing devices, and across a variety of cultures in real-time. We see it being used in a myriad of scenarios from passerby interacting with an installation via cellphones, to a medium-scale performance in an amphitheatre using both remote and local audience input as a layer in their composition, to large groups of asynchronous strangers weaving a complex mesh of beats together purely online. Using feedback from the launch, multiple live performances, and some viral attention, we're already planning the next iteration of the software that will alleviate some of the usability and accessibility issues as well as make the performative aspects even more engaging.

4.1 Evaluation

Since the initial launch, we've performed multiple early stages of case studies. Our initial test case, a small group of students in a classroom setting, revealed some inconsistencies with the interface and, more chronically, issues with our first set of instrument samples.

Further exposure and feedback through a post on Hacker News helped us to iterate and create a much more pleasant and engaging sonic experience by reworking the mixing, the samples themselves, and the way notes are birthed and aged. Finally, through a live on-stage performance to an audience of around 100 people, we identified new features that we intend to implement in the near future.

4.2 Future Work

We have already begun security improvements by significantly modifying our Twitter oAuth feature to be much less aggressive over permissions. We also plan to address some usability and predictability concerns by allowing users to preview what a note sequence might sound like before sending the tweet. Lastly, we plan to implement much more robust filtering features through a re-write of the sound engine. This will enable us to add effects to the overall sequence rather than the current note-by-note basis.

5. REFERENCES

[1] Dan Stowell. Supercollider sc140. `http://supercollider.sourceforge.net/sc140/`, October 2009. (Visited on 04/02/2015).

[2] Thor Magnusson. ixi lang: a supercollider parasite for live coding. *Proceedings of International Computer Music Conference*, pages 503–506, 2011.

[3] Jason Freeman, Sang Won Lee, Shannon Yao, and Aaron Albin. Lolc for laptop music ensemble. In *Proceedings of the 8th ACM Conference on Creativity and Cognition*, C&C '11, pages 433–434, New York, NY, USA, 2011. ACM.

[4] Joseph A. Paradiso. Electronic music: New ways to play. *IEEE Spectr.*, 34(12):18–30, December 1997.

[5] Jorge Herrera. The horgie: Collaborative online synthesizer, 2009.

[6] Nathan Weitzner, Jason Freeman, Yan-Ling Chen, and Stephen Garrett. massmobile: towards a flexible framework for large-scale participatory collaborations in live performances. *Organised Sound*, 18:30–42, 4 2013.

[7] Álvaro Barbosa. Displaced soundscapes: A survey of network systems for music and sonic art creation. *Leonardo Music Journal*, 13:53–59, 2003.

[8] Peter Gregson Daniel Jones. The listening machine. `http://thelisteningmachine.org/`. (Visited on 01/12/2015).

[9] Alexis Kirke and Eduardo Reck Miranda. A survey of computer systems for expressive music performance. *ACM Comput. Surv.*, 42(1):3:1–3:41, December 2009.

[10] Trygve M. H. Reenskaug. Mvc, xerox parc 1978-79. `http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html`. (Visited on 01/19/2015).

[11] Heroku. Object modeling in node.js with mongoose | heroku dev center. `https://devcenter.heroku.com/articles/nodejs-mongoose`, November 2014. (Visited on 01/19/2015).