CrowdPatching: a system for audience participation in algoraves

Jamie Beverley
McMaster University
beverljw@mcmaster.ca

David Ogborn
McMaster University
ogbornd@mcmaster.ca

ABSTRACT

CrowdPatching is an algorave performance system that leverages web technologies and interactive audio-visual components to facilitate audience participation. Audience members connect to a web application on their mobile devices that uses accelerometer data to generate values indicative of their individual and collective levels of *motion* and *coherence* with each other. The web interface also provides connected audience members with sliders to set their preferred levels for the high-level output parameters of the audio and visuals. The performer writes code to connect audience inputs (including *motion* values, *coherence* values, and slider levels) to the system's high-level outputs that drive generative audio and visuals. *CrowdPatching*'s live coding API and visuals are designed to clearly articulate to the audience how the performer is mapping their inputs to the outputs, and the system's participatory elements allow audience members to respond to the live coder's intentions. The live coder in a performance of *CrowdPatching* is thus primarily concerned with experimenting with how the audience reacts to their mappings and fostering interplay between themselves and participating audience members. A three layer parameter mapping strategy is employed in an attempt to create a structured and coherent audio-visual piece while allowing audience members to make meaningful contributions, and in the most extreme case bestowing complete control of the audio-visuals to the audience. The tradeoffs of performance spontaneity to musical coherence and system safety in the context of an algorave are discussed.

1 Introduction

CrowdPatching is an audio-visual performance system for audience participatory algoraves. Audience members connect to a web interface on their mobile devices where they are able to adjust faders to modify high-level parameters of the audio and visuals. Upon connecting to the web interface, each audience member has an avatar spawned in a 3D game world containing the visuals that are projected on stage. Values indicative of the audience' collective *coherence* and individual and combined levels of *motion* are determined by the system and are available to the live coder to be mapped to the audio and visuals. High-level output parameters of the system are controlled by the audience and performer in proportions determined by the *coherence* of the audience; a 25% *coherent* audience has 25% control over the output parameters, leaving 75% for the performer. In the most extreme case, a 100% *coherent* and synchronized audience can take complete control over the high-level parameters of the piece and drastically reduce the performer's ability to influence the audio and visuals. The performer writes code to map collective and individual audience inputs to the system's output, set their portion of the high layer output parameters, make harmonic, melodic, and rhythmic changes, and trigger sound events on audience members' devices. This work considers the tradeoffs between musical coherence, spontaneity, and audience participation, and experiments with the interplay between audience and performer in an algorave performance.

1.1 Motivations

CrowdPatching is primarily motivated by two goals. Firstly, *CrowdPatching* experiments with new performance objectives for the live coder. *CrowdPatching* challenges the performer to write code to affect the way in which the audience participates, rather than live coding generative audio or visuals. Secondly, *CrowdPatching* explores the challenge of creating unified sonic and visual results while simultaneously giving each audience member a critical and perceived role in the construction of the piece. Participatory pieces that allow audience members more control over the output risk losing structure or aesthetic quality due to the conflicting intentions of multiple contributors, while pieces that restrict audience participation in favor of structure may fail to convince audience members that their contributions are meaningful (Freeman 2005). *CrowdPatching* attempts to match the feeling of participation with significant sonic and visual output while striving for a unified sonic result.

Several recent performance pieces have explored the use of mobile devices for imagining new performer-audience relationships. *Fields* is an audience participatory piece in which the performers trigger sample playback and Web Audio

synthesis functions on audience members devices, creating a unique array of loudspeakers (Shaw, Piquemal, and Bowers 2015). *Crowd in C[loud]* is a piece in which the performer live codes changes to the mobile web instruments used by audience members, reimagining the role of the live coder to "live code the audience participation" (de Carvalho Junior, Lee, and Essl 2016). Live-coding duo Off<>zz's piece *Guide Our Glance* asked audience members to use their phones to vote on when the performers should look at each other, often instigating a movement in the progression of the piece (Noriega and Veinberg 2016).

CrowdPatching continues exploring the use of mobile devices for audience participation in the specific context of a live coding algorave performance. While algorave events do not have a consistent or clearly defined format, music and dancing are commonly accepted as imperative or at least desired (Collins and McLean 2014). The *CrowdPatching* system provides the live coder with tools other than just audio and visuals to encourage (or dissuade?) an audience from dancing. Audience inputs are mapped to audio and visual outputs by the performer explicitly through code, with the goal of eliciting different reactions from audience members and facilitating theatrical interplay between the audience and the performer.

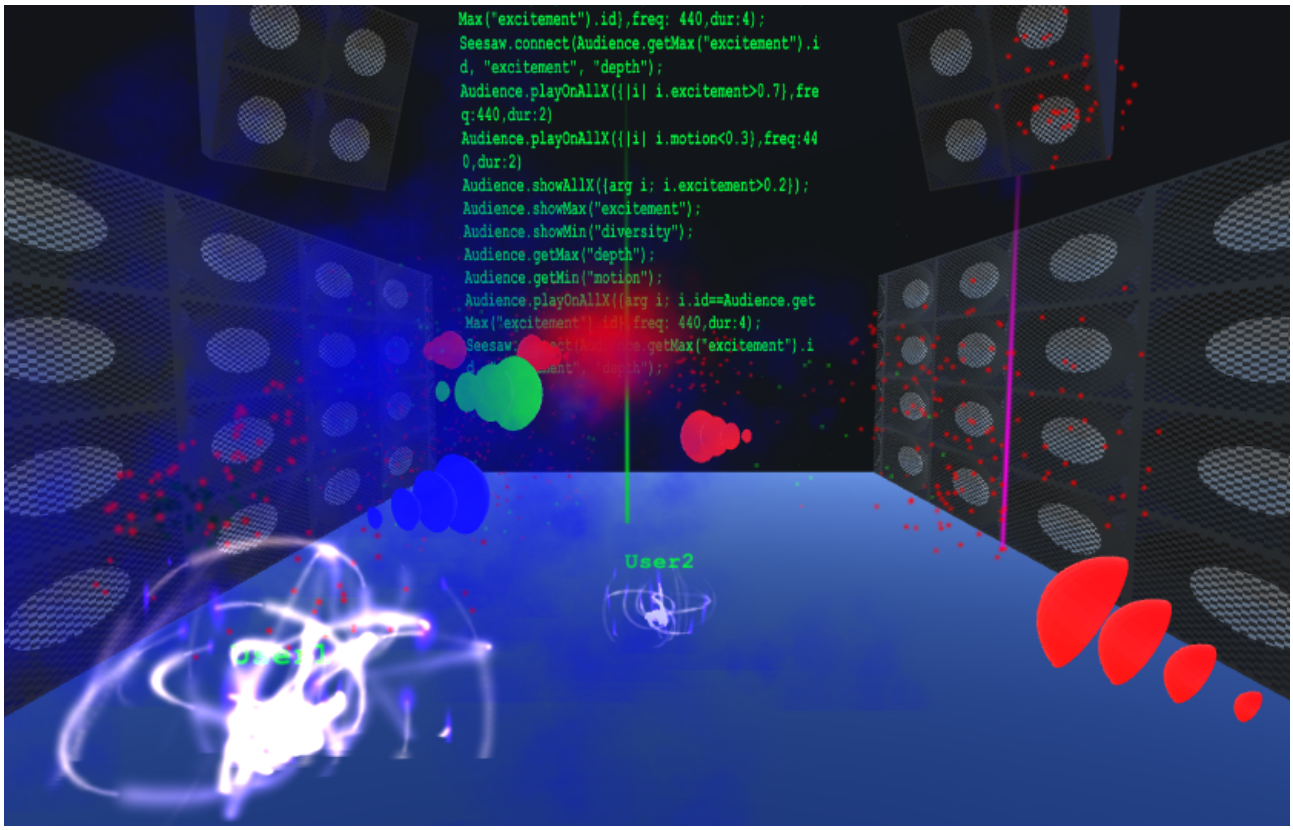

Figure 1: *Screenshots of CrowdPatching visuals with 2 audience members.*

2 Design

2.1 Mapping

CrowdPatching employs a three layer mapping strategy for mapping parameters to synthesis engines described by Hunt and Wanderley (2002). Normalized input parameters to the sound system are combined with other inputs to generate an intermediary input parameter (layer one) that holds a semantic definition (such as 'motion' and 'coherence'). Semantic intermediary output parameters are similarly mapped in a one-to-many relationship to various low-level outputs of the system (layer three). Layer two mappings connect semantic input layers to the semantic outputs.

In *CrowdPatching* the semantic input properties of the system were chosen to be audience *motion* and *coherence* to satisfy a desire for audience participation to be physical (suitable to an algorave). Individual audience member *motion* values are calculated by weighted-averaging scalar vectors of accelerometer data generated by their mobile device over four different time windows (0ms, 2500ms, 5000ms, and 10000ms) with greater weights attributed to longer windows. *Motion* values representative of the whole audience are calculated by averaging the *motion* values of all audience members. *Coherence* values are calculated by low pass finite impulse response filtering the variance between individual audience *motion* values at four different window sizes and then averaging the resultant four values.

The high-level output parameters are *diversity, depth,* and *excitement*. Broadly speaking, greater values of *excitement* correspond to faster rhythms, increased volume, and brighter tones, greater values of *depth* correspond to lower frequencies, more synthesizer grit, and more reflective reverberation, and higher *diversity* values correspond to greater pitch and rhythmic variation, and more dramatic panning movement. The instruments include a kick drum, a frequency-modulated bass, a snare drum, a hi-hat, an arpeggiator, a pad synthesizer, and web audio synthesizers run on audience devices. The scripts running the visuals translate greater *excitement* values to faster movement of game world objects, more elastic collisions between objects, and stronger magnetic force exerted on all synth avatars towards the camera, and greater *diversity* values increase the variation in color and direction of movement of synth avatars, while *depth* values control the distance of the back wall of the virtual room from the camera.

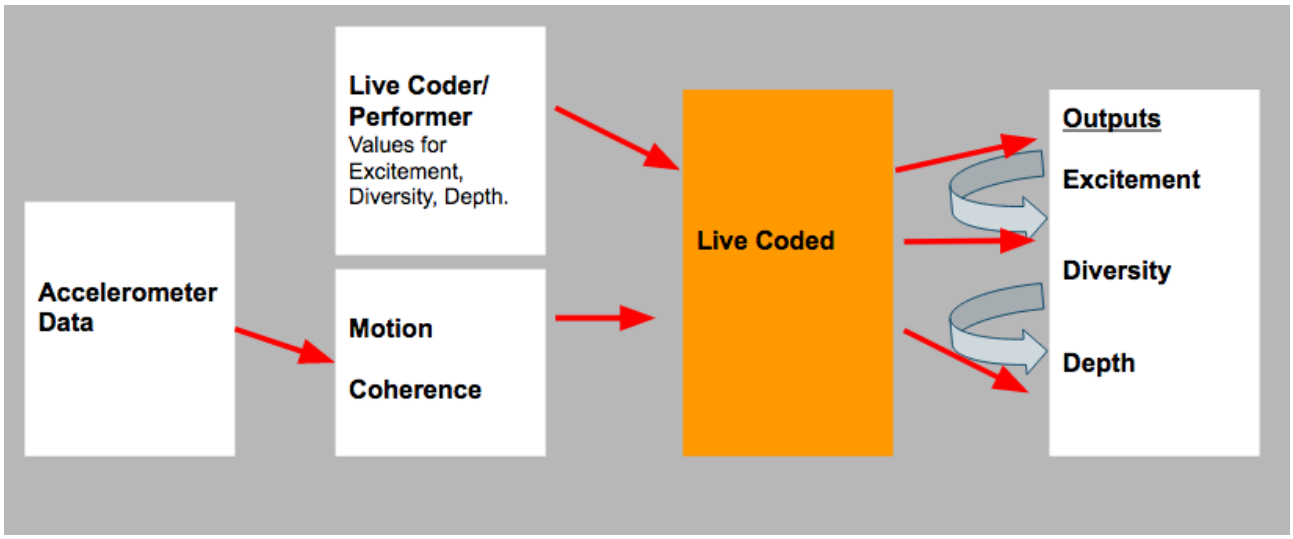

Figure 2: *Layer 3 Mapping in CrowdPatching.*

In *CrowdPatching*, three layer mapping allows the performer to bestow control of the high-level output parameters entirely to audience members with reduced risk that the audio and visuals will lose their algorave-inspired aesthetic and structure. The system leverages this advantage of three layer mapping in an attempt to give audience members a critical role while simultaneously creating art that strives to be satisfying to everyone (or at least pleasant to most). The extent to which the *diversity, depth,* and *excitement* are controlled by the performer versus an average of all audience members is dependent entirely on the *coherence* of the audience; the audience can take full control over the three parameters if they are completely synchronized. Furthermore, the system allows the performer to delegate 'solos' to individual audience members where they are granted complete control over the *diversity, depth,* and *excitement* of the system through controlling faders on the mobile interface. While not every audience member will be given a solo during a performance of *CrowdPatching*, perhaps demonstrating the feature will convince audience members of their potential to influence the piece.

2.2 Live Coding API

CrowdPatching's live coding API challenges the performer to experiment with theatrical interplay with the audience, rather than directly creating algorithms that control the low-level features of the audio and visuals (such as rhythms, pitches, effects, etcetera). The live coder is primarily tasked with connecting individual or collective audience inputs to the system's different output parameters (essentially live coding the layer two mappings in Hunt and Wanderley's model), and programmatically triggering sound events on targeted audience members' devices, while the audio and visuals react to the inputs provided by the performer and audience members.

The syntax of the live coding API (*figure 3*) attempts to explicitly indicate how audience inputs are mapped to system outputs, to allow audience members to comprehend and respond to the performer's actions. For instance, the performer could attempt to encourage (or dissuade) a cohort of the audience to move more by mapping their *motion* values to the *excitement* output parameter. The decision to adopt a transparent and conversational programming style is intended to foster the type of *public reasoning* Rohrhuber et al. (2007) observed to be characteristic of many live coding performances as a result of code sharing. In a performance of *CrowdPatching*, code sharing makes the commonly opaque action of parameter mapping public and more transparent to the audience, while the mechanism of participation invites audience members to respond to the performer's intentions in ways that directly affect the piece.

Code	Description
`Audience.playOnAllX({|i| i.excitement>0.7},freq: 440,dur:2)`	Plays note on all audience members devices with excitement greater than 0.7
`Audience.playOnAllX({|i| i.motion<0.3},freq: 440,dur:2)`	Plays note on all audience members devices with motion less than 0.3.
`Audience.showAllX({arg i; i.excitement>0.2});`	Will print user names and Id's of all audience members with excitement greater than 0.2.
`Audience.showMax("excitement");`	Prints username of audience member with greatest 'excitement' parameter
`Audience.showMin("diversity");`	Prints username of audience member with greatest 'diversity' parameter
`Audience.getMax("depth");`	Returns an Audience object of the member with the greatest 'depth' parameter
`Audience.getMin("motion");`	Returns an Audience object of the member with the least 'motion' parameter
`Audience.playOnAllX({arg i; i.id==Audience.getMax("excitement").id},freq: 440,dur:4);`	Plays a note on the audience member with the greatest excitement
`Seesaw.connect(audienceID:1, audienceParam:"excitement", performerParam:"depth");`	Connects the audience member with ID 1's excitement parameter to the performer's depth parameter

Figure 3: *CrowdPatching Live Coding API.*

2.3 Visuals

The visuals and code sharing in *CrowdPatching* are contained in a 3D game world implemented with the Unity Game Engine. The game world is intended to appear as an extension of the wall on which it is projected, and is stylized to match the aesthetic of an algorave. Every time a new synth is triggered, the game world spawns a game object that exists for the duration of the note. When audience members connect to the mobile interface an avatar with their username is spawned in the scene. Audience members can move their avatar by flicking their device in the (x, y, z) direction they would like to move, absorbing any synth objects they collide with. Apart from being a mildly entertaining way to interact with the visuals, the virtual positioning of participating audience members 'onstage' perhaps helps disassemble typical performer-audience power dynamics that accompany onstage performances. The live coding group Powerbooks Unplugged disperse amongst the audience instead of performing on stage partly for this purpose (Rohrhuber et al. 2007). Disrupting typical performer-audience power dynamics by virtually placing audience members on stage may contribute to fostering a space of public reasoning; the performer no longer completely controls all visual communications, potentially opening up room for audience members to take more control.

2.4 System Architecture

Audience members connect to a web interface served by a NodeJS application. A web interface sends accelerometer data to the NodeJS server and listens for messages to trigger Web Audio synthesis functions. A NodeJS server process generates *motion* and *coherence* values and relays them to a SuperCollider script running generative audio patterns that respond to the inputs, and the commands of the live coder. SuperCollider also sends messages back to the NodeJS server to distribute events that trigger Web Audio synthesis functions on audience devices and relays messages to Unity which produces the generative visuals. (see *figure 4*)

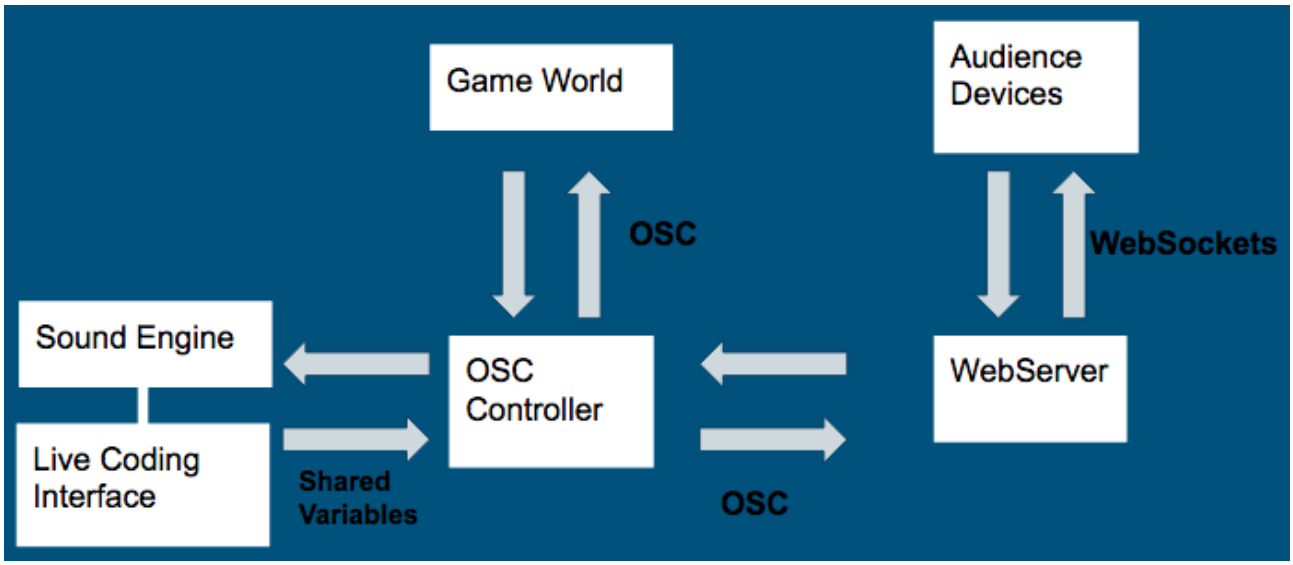

Figure 4: *CrowdPatching system architecture.*

3 Discussion and Future Work

Freeman (2005) posited that large participatory pieces cannot make each audience member feel essential to the composition while simultaneously satisfying everyone's tastes. *CrowdPatching* attempts to meet both demands at the expense of imposing restrictions on the output. While an audience member may be given complete control over the high-level parameters, the performer still sets the boundaries on what the maximum and minimum levels of *diversity, depth,* and *excitement* produce, prior to the performance. Audience members may be given complete control at the expense of narrowing the range of musical and visual possibilities afforded by the system.

The tradeoffs of performance spontaneity for musical and visual coherence in *CrowdPatching* may be of broader significance to the field of live coding. Cox (2015) noted that the uncertainty inherent to the live manipulation of complicated algorithms allows live coding practice to be a novel non-goal-oriented research methodology. Collins and McLean (2014) similarly described algoraves as 'experiments' where failures are learned from and potentially embraced. Future iterations of

CrowdPatching may benefit from Paz (2015)'s methodology for high-level features rule generation for live coding performance, which proposes some strategies for reincorporating variability into pre-programmed algorithms.

While the *CrowdPatching* system permits the creation of new patterns, the predefined generative algorithms are designed to stand on their own, freeing the performer to live code other aspects of the performance. In exchange, the live coder is tasked with finding creative ways to work with many high-level inputs. One proposed technique is to elicit audience members to participate in a certain way by clearly articulating how their inputs are being mapped to the system. Future performances of *CrowdPatching* will challenge the performer to experiment with other new strategies for engaging with the audience.

The graduate research of the first author will continue exploring audience participation in live coding with a pedagogical focus. Future iterations of the system will benefit from implementing more advanced algorithms (particularly machine learning algorithms) for generating values indicative of audience *motion* and *coherence*, and incorporating feedback from audience members.

4 Acknowledgements

I would like to thank Dr. David Ogborn for his generous support and guidance, the McMaster Arts and Science Program and Dr. Jean Wilson for providing me the opportunity to complete this work, and the three anonymous reviewers who provided valuable comments and feedback.

5 Links

Demonstration Video: https://youtu.be/mn8cr1iEqLQ

Code Repository: https://github.com/JamieBeverley/CrowdPatching

References

Collins, Nick, and Alex McLean. 2014. "Algorave: A Survey of the History, Aesthetics and Technology of Live Performance of Algorithmic Electronic Dance Music." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 355–58. London, United Kingdom: Goldsmiths, University of London. http://www.nime.org/proceedings/2014/nime2014_426.pdf.

Cox, Geoff. 2015. "What Does Live Coding Know?" In *Proceedings of the First International Conference on Live Coding*, 170–78. Leeds, UK: ICSRiM, University of Leeds. doi:10.5281/zenodo.19329.

de Carvalho Junior, Antonio D., Sang Won Lee, and Georg Essl. 2016. "Understanding Cloud Support for the Audience Participation Concert Performance of Crowd in c[loud]." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 16:176–81. 2220-4806. Brisbane, Australia: Queensland Conservatorium Griffith University. http://www.nime.org/proceedings/2016/nime2016_paper0037.pdf.

Freeman, Jason. 2005. "Large Audience Participation, Technology, and Orchestral Performance." In *Proceedings of the International Computer Music Conference 2005*.

Hunt, Andy, and Marcelo Wanderley. 2002. "Mapping Performer Parameters to Synthesis Engines." *Organised Sound* 7 (2). Cambridge University Press: 97–108.

Noriega, Felipe Ignacio, and Anne Veinberg. 2016. "Guide Our Glance." Lecture Recital. In *Proceedings of the International Conference on Live Coding 2016*.

Paz, Ivan. 2015. "Live Coding Through Rule-Based Modelling of High-Level Structures: Exploring Output Spaces of Algorithmic Composition Systems." In *Proceedings of the First International Conference on Live Coding*, 83–86. Leeds, UK: ICSRiM, University of Leeds. doi:10.5281/zenodo.19326.

Rohrhuber, Julian, Alberto de Campo, Renate Wieser, Kees van Kampen, Echo Ho, and Hannes Hölzl. 2007. "Purloined Letters and Distributed Persons." In *Proceedings of Music in the Global Village 2007*.

Shaw, Tim, Sébastien Piquemal, and John Bowers. 2015. "Fields: An Exploration into the Use of Mobile Devices as a Medium for Sound Diffusion." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, edited by Edgar Berdahl and Jesse Allison, 281–84. Baton Rouge, Louisiana, USA: Louisiana State University. http://www.nime.org/proceedings/2015/nime2015_196.pdf.