ACCRETION: FLEXIBLE, NETWORKED ANIMATED MUSIC NOTATION FOR ORCHESTRA WITH THE RASPBERRY PI

K. Michael Fox
Rensselaer Polytechnic Institute
kmichaelfox.contact@gmail.com

ABSTRACT

In 2014, the author set out to expand the notational potential of their generative music systems to be performed by the Rensselaer Orchestra in Troy, NY. The experiments resulted in the use of several networked Raspberry Pi devices delivering a realtime, generative Animated Music Notation to subsections of the live orchestra during performance. This paper outlines the structure of the piece, *Accretion*; the technical details of its implementation; and the possibilities presented by using the Raspberry Pi to deliver scored materials to performers. Ultimately, the paper seeks to make a case for adopting the Raspberry Pi as a powerful device and method of distribution/performance of Animated Music Notation.

1. INTRODUCTION

This paper describes the author's composition for orchestra, *Accretion*, and the technological developments that enabled the creation and performance from an Animated Music Notation (AMN). *Accretion* is structured around techniques and structural schemata derived from granular synthesis. The term accretion refers to the formation of a thing by means of some attraction, perhaps gravitational as in the case of the formation of celestial bodies like planets, stars, and nebulae. This process works as a sufficient metaphor for the kinds of interactions, including granular synthesis, that the author has previously implemented in electronic and electroacoustic compositions. With granular synthesis, sounds that are near imperceptibly short can be overlapped in very high densities to produce particular sonic textures to emerge. The piece *Accretion* is a translation of these electronic idioms into the context of the acoustic orchestra.

The realtime generative systems comprising my previous work have diverse inspirations, origins, and implementations. However, *Accretion* explicitly seeks to incorporate a previously unused feature: rendering the internal interactions of the generative system into scored material that is performed in realtime by an ensemble. The opportunity to write for an orchestra posed that significant challenge in realtime notation. In order to translate these techniques from the realm of microsounds to the timescales and context of the orchestra, a custom AMN was designed. I set about to design a system that used the kinds of compositional methods I enjoy in the digital realm and rendered them legible (and with the utmost specificity) to the orchestra.

Gerhard Winkler, describing his own work towards realtime generative notation for ensembles, has emphasized the benefit of removing as much of the simulation hardware from the stage as possible [1]. However, it was clear that using a single screen was neither ideal nor practical. The projected image, for an ensemble the size of an orchestra, would have have to be impractically large to legibly display all necessary player parts. Additionally, the projection would either have to be positioned behind the audience, or the orchestra would have to face away from the audience. The solution I developed divided the score into four different screens; networked and synchronized these screens with a master simulation; and, positioned these screens in such a way that was unobtrusive to the audience while remaining legible to players up to 10 meters away. With these challenges in mind, I consulted other works using networked devices, screens or processes, including those described by Winkler [1], Jason Freeman's LOLC [2], and Decibel Ensemble's "Score Player" [3]. The Raspberry Pi (RPi) was affordable and flexible enough as a platform to realize these goals, and I adopted the device as a way to enable the compositional goals I had established. The rest of this paper describes the process of moving from compositional intent to functional Animated Music Notation design, its technological implementation, and future trajectories for the system.

2. *ACCRETION*

My compositional intentions with *Accretion* required each section of the orchestra to act independently of the others, facilitating the coordination of instrumental articulations into clouds where each event had a seemingly arbitrary timing. The components and structure of the piece, being derived from granular synthesis, relied on events happening in absolute time, as opposed to subdivisions of metrical time based on tempo. Coordination of these events, then, form clouds or clusters that are partially identified by their densities. However, since the instruments playing these sound grain-like notes are resonating bodies activated by humans, there are three additional components introduced: pitch, playing technique and articulations, and dynamics. Together, these components formed the main design considerations of the simulation system, programmed in C++ with openFrameworks.

2.1 Time & Pitch

The generative software system created "events" of two types: singular or durational. The singular events were realized as the shortest possible articulation of a note the instrument. Durational events, on the other hand, could be long sustained notes or collections of staccato notes occurring in a strictly defined time duration. Both event types consisted of a single pitch assigned at the time of generation. These pitch assignments are determined by an active "global" pitch class, constraining all instrumental parts to a pre-defined harmonic space, specifically the octatonic (WH) scale and the whole-tone scale.

Durational notes had an additional property when comprised of collections of short notes. These staccato notes were to be articulated as fast as possible at the prescribed dynamic. The resulting effect is a slightly asynchronous timing for the events resulting from the mechanical nature of the action and the limitations of the human body. This led to subtle emergent variation on the overlapping patterns of coexistent events, which was further amplified by the use of different playing techniques.

2.2 Technique

For each player part, divided by instrument sections, the events were assigned articulation techniques idiomatic to the instruments. Because the orchestra was comprised of student players, I limited the techniques to those that they would feel comfortable and confident performing (i.e., not extended). A string instrument could perform events as one of: arco, pizzicato, col legno tratto, col legno battuto, staccato, or tremolo; while winds would more uniformly play events legato.

These techniques would be assigned to each event as that event was generated and could apply to different event types in different ways. For example, col legno battuto for a singular event would be realized as a single strike of the bow, while the durational event using col legno battuto would be realized as multiple strikes of the bow over the course of the specified time duration and articulated as fast as possible.

2.3 Dynamics

In the case of singular events, a single discrete dynamic is generated (as there is only one short note played). Durational events, however, are assigned continuous dynamics that vary over the time duration of its articulation. Whether the durational event is a sustained note or a collection of short notes, these are contained within a the continuous dynamic envelope that makes each moment of that event vary in volume, intensity, and timbral quality.

Dynamic envelopes for these durational events were based on the Attack, Decay, Sustain, and Release (ADSR) envelopes of electronic sound synthesis. However, abstracted from the synthesis function, each phase of the envelope can increase or decrease an Attack phase of an envelope need not start at zero and can decrease before encountering the beginning of the Decay phase. The one exception to this freedom is the Release phase, which will always approach zero at the end of its duration.

These envelopes are applied to the durational events to vary the dynamics at any given moment between *dal niente* (when possible on the instrument) and *fff* (available, but seldom reached). Since dynamic envelope is given to each durational event, each section of the orchestra is completely decoupled from the others with respect to crescendi or descrescendi. This allows the ebbing and flowing of different timbres over and under each other in graceful coordination (or, in reality, lack thereof).

2.4 Notational Framework

David Kim-Boyle has noted that "computer-generated scores, particularly those that employ real-time animation, create a heightened sense of anticipation amongst performers, especially given that performance directive can change from moment to moment" [4]. Similarly, Pedro Rebelo notes that there is a delicate balance in animated notations between representing gestures too literally and too abstractly [5]. Given these two considerations and the goals of the piece, I believed that it was important to render the notation in a form that was reasonably approachable by any of the performers. Like the use of idiomatic over extended techniques, I wanted to present the notation for my piece which functionally achieved the specific timings and re-

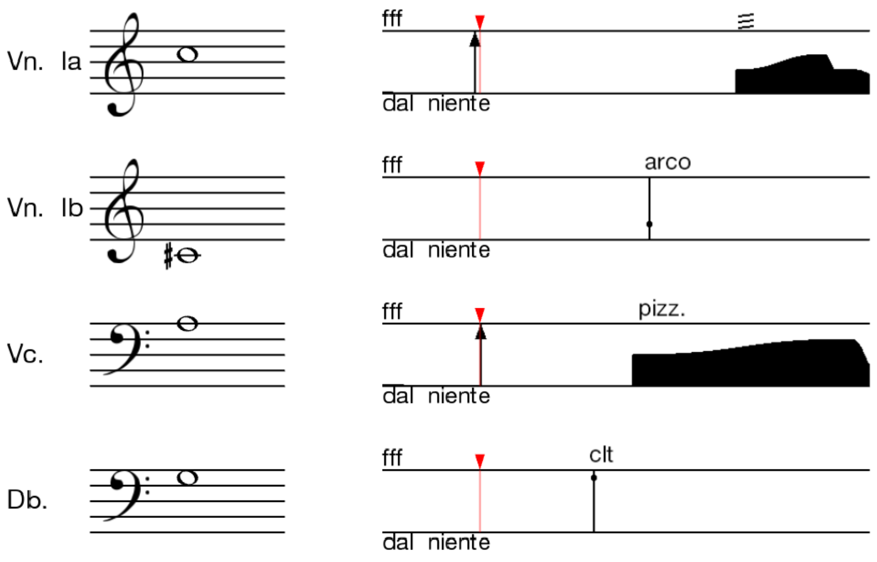


Figure 1. Reduction of concert score.

sults that I desired while taking the form of notation that the student players would feel confident reading and performing from. This led to an extension of conventional notation with meaningfully animated gesture-figures. As seen in Fig. 1, on the left side of the score, notes for each instrument's *next* or *current* event were displayed on staves with accidentals. To the right, the articulation for each instrument's gestures would slide from right to left down a pipeline.

Using the "playhead" style indicator described by Hope and Vickery, the articulation point occurs when a gesture crosses a red line with a downward facing arrow [6] (See Fig. 3). For singular events, the dynamic of the articulation is featured as a small circle that is vertically bisected by a black line (See Fig. 4). The performers were coached to regard the center of this circle both as the "dynamic value" of the note and as the point of attack as it passed through the playhead indicator. Thus, as the circle passes the playhead a note at the pitch specified on the staff is articulated with the specified technique (described below) at the dynamic corresponding to the height of the circle's center point. For durational events, the dynamic is read as the height of the top of the envelope at the point where it is currently intersecting the playhead. In either case, the vertical range of the envelope pipeline is listed on the score as *dal niente* at the bottom and *fff* at the top. Most instruments were expected to perform *dal niente* as the quietest dynamic they could possibly play.

Since all durational events approach *dal niente* as they conclude, the final moment of these types of events was initially very ambiguous. Because of the specificity that I was seeking with the score, I added a vertical line to clearly demarcate the termination point of these envelopes. To distinguish the function of this line from the line that is connected to singular events, the termination point featured an upward-pointing triangle at its top that is clearly distinguishable from the circle (of the singular event). Similar to the singular event, the performers were coached to interpret the top-most point of the triangle as the point of

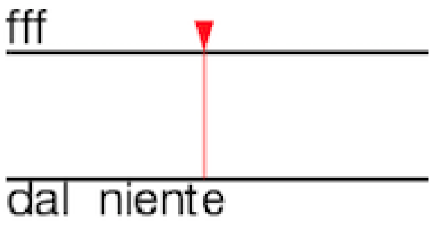


Figure 2. Playhead notational indicator.

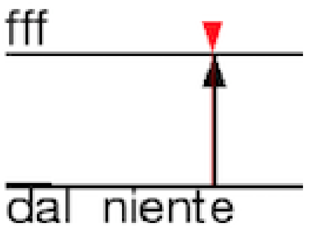


Figure 3. Termination of durational event.

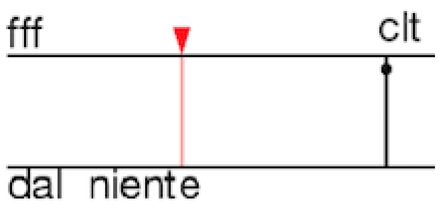


Figure 4. Singular Event.

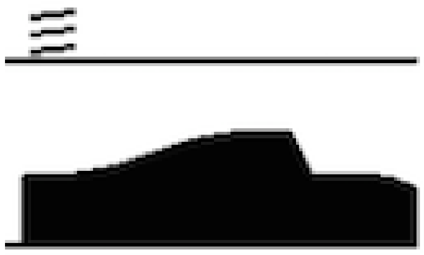


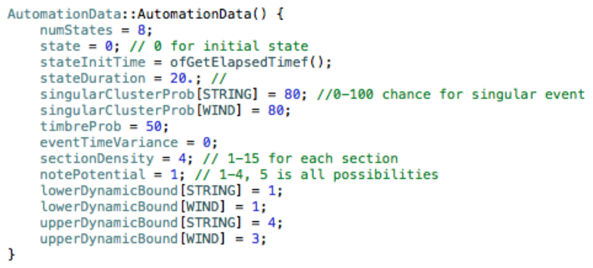
Figure 5. Durational event envelope.

contact as it passes the playhead. This helped to cue events such as that pictured in the Fig. 3, where a part stays at *dal niente* for a long period of time (seemingly absent).

Initially, the technique was placed above the notes on the stave, as would be expected on a more traditional paper score. However, rehearsals demonstrated that much more consistent attention was required for the dynamic envelopes, and the techniques were more effectively executed when they were rendered with the envelope to which they referred. This resulted in envelopes that are rendered with their respective technique also moving down the pipeline directly above their leading edge.

3. RASPBERRY PI AS NOTATION RENDERING CLIENT

The most critical point in understanding the notation rendering system is the distinction that the notation itself is a front-facing, or front end notational representation of events generated within a simulation system, or back end. To understand how the client renders these events, I will first describe how the simulation system generates events within *Accretion*. At a macro-level, the piece is organized around clusters of events with particular attention to densities of players, the types of events they are performing (and in what distribution the types of events appear amongst these active players), and the dynamic levels of these events. Within these events, or clusters, the collection of events are generated much like random grains are spawned within granular synthesis clouds. Fig. 6 shows a C struct called automationData that holds the distribution parameters of the event generation. As the piece progresses, these distribution parameters are updated to vary the density, number



```
AutomationData::AutomationData() {
    numStates = 8;
    state = 0; // 0 for initial state
    stateInitTime = ofGetElapsedTimef();
    stateDuration = 20.; //
    singularClusterProb[STRING] = 80; //0-100 chance for singular event
    singularClusterProb[WIND] = 80;
    timbreProb = 50;
    eventTimeVariance = 0;
    sectionDensity = 4; // 1-15 for each section
    notePotential = 1; // 1-4, 5 is all possibilities
    lowerDynamicBound[STRING] = 1;
    lowerDynamicBound[WIND] = 1;
    upperDynamicBound[STRING] = 4;
    upperDynamicBound[WIND] = 3;
}
```

Figure 6. “automationData” struct.

of active players, the probability of each instrument family generating events, and the probability of which event types will be active at any given point.

Following from the notion that the score is a front-facing representation of the system’s event generation, the networked RPi scores are simply the interface elements of the simulation without the simulation backend running locally on the devices. Instead, the RPi’s listen for OSC messages that encapsulate the parameters of events as they are generated. At the moment when an event is generated, its parameters are packaged as this OSC message and broadcast to the clients via a LAN connection. The address of these messages takes the format “\number_of_RPi_device\ number_of_part” and these address numbers are hardcoded locally. These messages take different forms depending on the type of event they represent, so each OSC message starts with an identifier of which event type it refers to, 0 for singular events and 1 for durational events. When the messages represent a singular event, the rest of the message’s arguments includes, in order: an integer midi pitch value, an integer delay in milliseconds from the start of the cluster, an enumerated integer value for the dynamic level, an enumerated integer value for the instrument family (to identify the technique behavior), and an enumerated integer value for the technique type (Winds only use legato, so 0 is used as a placeholder).

When a durational event is created, the OSC message must describe the parameters of the ADSR envelope. The arguments that appear after the durational events first identifier argument are as follows: an integer midi pitch value, an integer delay in milliseconds from the start of the cluster, an integer value for Attack time in milliseconds, an enumerated integer value for initial Attack dynamic, an enumerated integer value for the final Attack dynamic, an integer value for Decay time in milliseconds, an enumerated integer value for the final Decay dynamic, an integer Sustain time in milliseconds, an enumerated integer value for the final Sustain dynamic, an integer value for Release time in milliseconds, an enumerated integer value for instrument family, and an enumerated integer value for technique type. The instrument identifier and technique type behave as described for the singular events.

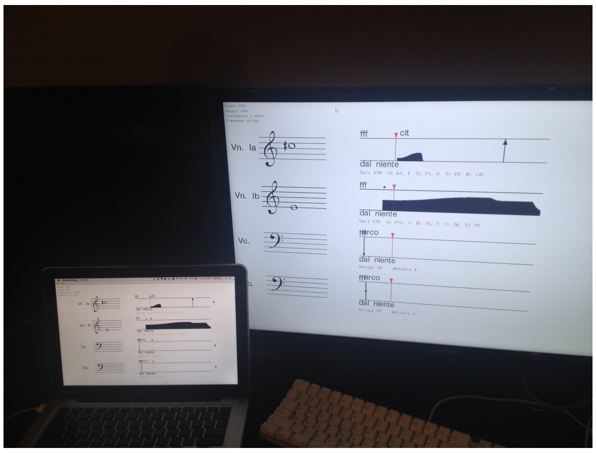


Figure 7. Score rendering on linked Server (laptop, left) and Client (large monitor, right) applications.

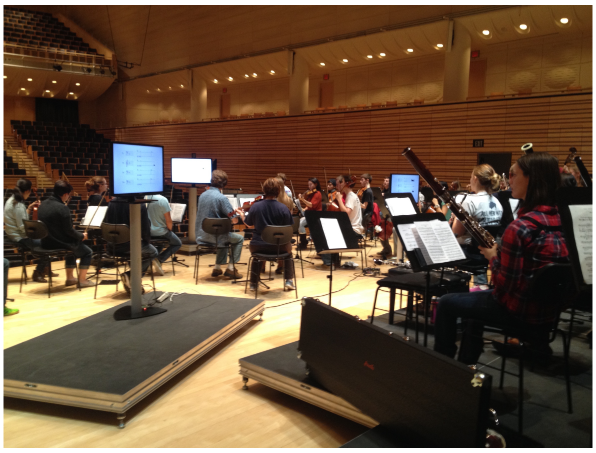


Figure 8. Stage setup in the Concert Hall at Rensselaer Polytechnic Institute's Experimental Media and Performing Arts Center. Monitors 2 (middle), 3 (right), and 4 (left) are visible.

Fortunately, openFrameworks is becoming increasingly well-supported for the Raspberry Pi. The code that rendered the notation could be written initially on an Apple OS X computer and redeployed later by recompiling the application for ARM devices (RPi). Minor performance optimizations were then be implemented to assure smooth performance on the different architectures. One such improvement was using a time-based tick system for progressing the system. This allowed for slight fluctuations in the framerate of the rendering system while maintaining the temporal specificity necessary for the coordination of events across multiple RPi's.

The HDMI support that is natively included with the RPi allows easy connection to most monitors or projectors on hand. In the case of *Accretion*, I used four 32 inch video monitors. Each of the RPi devices are small enough that they could be located on stage with the monitors displaying the rendered material and supplied only with power and ethernet cables. Thus, each monitor worked as one client in a vaguely server-client relationship with the simulation. Because the RPi works just like most other Unix systems, the rendering client on each device could be activated using SSL from the simulation machine off-stage at the beginning of the performance. Once the performance was over, each device was sent kill message also using the SSL connection, allowing a quick strike from the stage.

4. CONCLUSION & FUTURE WORK

The first major benefit that has emerged since completing *Accretion* is the portability of the score. These inexpensive RPi devices are small enough that they can easily be shipped to new ensembles, plugged in, and ready for rehearsal or performance in a relatively short amount of time. By supplying the devices themselves, instead of the application or source files for compilation, this setup has incidentally avoided the potential headaches stemming from incompatible versioning of libraries or system architectures.

Furthermore, the RPi features a number of standard GPIO ports that allow a variety of sensors to be attached. Each Raspberry Pi client could simultaneously collect sensor data, communicate that data back to the server system via OSC, and create responsive feedback loops for interactive generative music systems with human performers. Though *Accretion* was not interactive or responsive to outside elements in the performance, this is fertile ground of future work and significant precedent exists for adaptive realtime scores [Nick Didkovsky's *Zero Waste*, Harris Wulfson's *LiveScore* system [7], Paul Turowski's *Hyperions*, Jason Freeman's LOLC system [2]].

Likewise, the devices are easily extended with other microcontroller hardware, such as Arduino boards. By attaching external Arduino devices to the device, it could be possible to do sophisticated distributions of sensor and data processing in a highly networked setting.

There is also flexibility in the spatially-distributed networking potential on these devices, enabling work reminiscent of Arthur Clay's *China-Gates* or *Going Public*. RPi's are equipped out of the box with ethernet capabilities, and are extendable with usb wifi adapters, a tremendous advantage in distributed or spatialized performance situations. For instance, it could be possible to synchronize devices in remote parts of a building via a network without any significant demand to setup or infrastructure.

All of these considerations, however, require extensions to hardware, software, notational practices, or all of the above. This seems to be the nature of much of the prominent practices in AMN already, though. Each composition's needs can be contextually defined to the extent that entirely new notational schemata might be created for individual pieces. Indeed, the system I designed was motivated by the need to realize the compositional ideas that formed *Accretion*. However, I see many potentials for generalizing the tool along several of the trajectories outlined above to further enable the compositional ideas I am interested in.

The main addition that I see for the system is inspired by another common usage of the RPi: the networked media browser. Common media browser applications include KODI (formerly XBMC). In the case of networked scores, this type of score access would require the development and implementation of configuration flat-files to aid in the networking functionality. The configuration file schema would theoretically allow scores to specify their role in the network at runtime (such as the particular instruments they display parts for, in the context of *Accretion* for example), or support preset values that can be implemented at device startup.

Similar benefits are represented by the ScorePlayer app, developed by the Decibel Ensemble. This implementation simply provides an alternative platform that enables a more diverse configurability and extended low-level control over notational structures, but at the expense of a less widely distributed device (relative to iOS devices and their ostensible ubiquity). However, as stated above, the packaging of scores on a standalone, pre-configured device, but requiring little to no setup by performers, may increase the potential for wider distribution.

Acknowledgments

Special thanks to Ryan Ross Smith and Nicholas DeMaison.

5. REFERENCES

[1] G. Winkler, "The realtime-score. a missing-link in computer-music performance," in *Sound and Music Computing*, Paris, 2004.

[2] S. W. Lee, J. Freeman, and A. Collela, "Real-Time Music Notation , Collaborative Improvisation , and Laptop Ensembles," in *NIME 2012 Proceedings of the International Conference on New Interfaces for Musical Expression*, 2012, pp. 361–364.

[3] C. Hope and A. Wyatt, "Animated music notation on the ipad." in *PROCEEDINGS OF THE INTERNATIONAL COMPUTER MUSIC CONFERENCE*, ser. International Computer Music Conference, 2013, pp. 201 – 207. [Online]. Available: `http://quod.lib.umich.edu/cgi/p/pod/dod-idx/animated-music-notation-on-the-ipad.pdf?c=icmc;idno=bbp2372.2013.025`

[4] D. Kim-Boyle, "Real-time score generation for extensible open forms," *Contemporary Music Review*, vol. 29, no. 1, pp. 3–15, February 2010.

[5] P. Rebelo, "Notating the unpredictable," *Contemporary Music Review*, vol. 29, no. 1, pp. 17–27, February 2010.

[6] C. Hope and L. Vickery, "Screen scores: New media music manuscripts." in *PROCEEDINGS OF THE INTERNATIONAL COMPUTER MUSIC CONFERENCE*, ser. International Computer Music Conference, 2011, pp. 224 – 230. [Online]. Available: `http://libproxy.rpi.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsbl&AN=CN080981465&site=eds-live&scope=site`

[7] G. D. Barrett and M. Winter, "Livescore: Real-time notation in the music of harris wulfson." *Contemporary Music Review*, vol. 29, no. 1, pp. 55 – 62, 2010. [Online]. Available: `http://libproxy.rpi.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=53772450&site=eds-live&scope=site`