The Dark Side

Shawn Lawson
Rensselaer Polytechnic Institute
lawsos2@rpi.edu

Ryan Ross Smith
SUNY Oneonta
ryanrosssmith@gmail.com

ABSTRACT

This paper describes the authors solution to the challenges of turning a no-distance collaboration into a long-distance one. The solution was The Dark Side of The Force, a new coding environment designed to meet the authors' needs. In this paper we describe what led to the development of The Dark Side, and how The Dark Side led to new artistic paths.

1 Introduction

Authors Shawn Lawson and Ryan Ross Smith have been collaborating as The Rebel Scum since 2013. Early works, including *Kessel Run*, (2014) and *Sarlacc*, (2015), were designed as standalone and repeatable audio-visual performance works featuring an early iteration of Lawson's The Force live-coding integrated development environment (IDE) for visuals.[1] With these two works the audio-visual relationship is baked into the production of the works and the works themselves in performance, but the inflexibility of this cohesiveness reduced the possibility for a satisfying collaborative link in live performance.

With *Owego System Trade Routes*, (2016), (*OSTR*) the authors attempted to improve the functional relationship between Lawson's visuals and Smith's audio, and to create a more satisfying performance environment by moving away from the more-or-less fixed formal structures of *Kessel Run* and *Sarlacc*. The introduction of a modular synthesizer and the development of the AppiOSC encouraged an element of performance indeterminacy and anxiety missing from the aforementioned works (Shawn Lawson 2016).

The AppiOSC, designed in collaboration with Frank Appio, enabled a two-way bridge between Lawson's code from The Force and Smith's modular synthesizer by converting control voltage into streams of numbers and vice versa. Perhaps most interesting was the method by which Lawson's number streams were created. In any text environment, especially the real-timeliness of live-coding, one is not generally concerned with the text beyond what it represents and prescribes. For instance, within the introduction to this paper the authors have been briefly outlining their past work, and hopefully these concepts have been represented with some degree of clarity. The concept of the AppiOSC is that the words represented are full of letters, spaces, and punctuation that enable us to clearly project our ideas but those letters, spaces, and punctuation marks are not the ideas in and of themselves. With the AppiOSC, the authors approached these elements as "leftovers" of the coding process to create an interesting strategy for generating dynamic numerical material that is likely not purposeful or meaningfully controllable.[2] In a sense, *OSTR* represents the authors intention to move away from predefined structure toward momentary stochasticity and narrative vulnerability by interpreting the code as a raw collection of symbols.

After completing a series of performances and recordings of *OSTR* with the AppiOSC, the practical realities of Lawson and Smith's long-distance collaboration made continued work impossible in its current form. The obvious solution was to develop some method of telematic collaboration that facilitated regular rehearsals and development sessions as well as introducing the possibility for displaced performance/presentation into the future. Despite its huge potential for crash-y, crackling and craze-inducing malfunctions, telematic performances are generally easy to organize with video chat programs like Skype or GoogleChat; and Chris Chafe's JackTrip still inspires with its high-fidelity and low-latency.

Still, because The Force is browser and text based, the authors conceptualized a new IDE capable of circumventing the traditional telematic transmission model, as the authors' primary requirement for an efficient displaced collaboration was reliable telematics for high-fidelity audio and visuals in rehearsal and performance.

[1] The Force - https://github.com/shawnlawson/The_Force

[2] i.e. how many Y's are in a particular code block. Additionally, one author found it amusing to purposefully insert unnecessary letters that modulated the control voltage output.

2 The Dark Side of The Force

The Force was originally designed as a graphics-only live-coding system for a single performer. The Dark Side is a ground-up rewrite of The Force in which the graphics middle-ware is replaced and including many new features, yet still maintaining the ease and familiarity of the original user interface (UI). The following sections briefly outline the primary features and implementation notes of The Dark Side.

2.1 Telematic

The Dark Side uses ACE, Firebase, and FirePad.[3] ACE is the text editor. Firebase is a Google owned real-time database. FirePad is a Javascript middle-ware layer that sits on top of ACE, runs client-side, and invisibly manages the multi-user concurrent editing. FirePad and Firebase are designed to work together, such that changes submitted to the database are immediately emitted to all listeners. In other words, every user who is listening will know immediately of any change made by any other user, not dissimilar to an off-the-shelf multi-user text editor like Google Docs. As we'll describe later, only textual changes are transmitted telematically, and all of the audio and visuals are rendered locally/client-side, resulting in low-latency and low-bandwidth (approximately 5-10 MB per hour-long rehearsal/performance).

2.2 Multi-language

The decision was made early on to support multiple languages in a single text editor buffer, in this case TidalCycles and OpenGL Fragment Shader. This was done to give equal footing to both audio and visual languages, clean-ness of the UI, and the possibility for creating shared variables. The latter is a continuing effort, but has not yet come to fruition as other serendipitous events, described later in this paper, led the authors down interesting creative paths.

It should be pointed out that there are already several multi-user IDEs in existence, including Extramuros and Overtone, IDEs that support both audio-visual, LuaAV and Extempore, and an IDE for both multi-user & audio-visuals, Gibber.[4] The only real distinction the Dark Side has in comparison to all of these is that it is multi-lingual.

2.3 Code Captured Performance

With significant inspiration from Sang Won Lee's text writer, the authors implemented code captured performance for the recording and play back of their work (Lee and Essl 2015a; Lee and Essl 2015b). This resulted in several findings similarly found by Lee and Essl. First, recordings of performances and rehearsals had no impact on computer performance and storage due to the minimal size of a code capture, approximately 10MB per hour. Second, due to how the code capturing was implemented, including multi-cursor movement, window scrolling, and tidal execution high-lighting, the personality of each performer comes through in the playback. Third, because the capture is of the code itself, there is no audio or visual compression of any kind, meaning that playback quality is limited only by the quality of the computer.

2.4 High-Fidelity

Because only the code is transmitted and audio & visuals are all rendered locally, both members of a telematic collaborative team experience full-quality audio and visuals, and this is extensible to more than just the authors' rehearsal and performance practice. With this system it is not necessary for the performers to be present at a particular venue while still producing full-quality audio and video. The venue simply sets up a computer that acts as an observer to the collaborative performance, "tapping" into the collaborative performance that is happening in one or more displaced locations.[5] At venues with unstable Internet connectivity, hot-spotting through a phone is possible due to the low-bandwidth requirements of The Dark Side. When considering the Location/Interaction chart of Lee and Essl, if all performers are remote from the venue computer, then it seems like sitting on the line between Co-Located-Synchronous and Remote-Synchronous is a place to start. The question becomes how to differentiate the venue's computer as performer or participant of the performance. It could be argued that the computer is not dissimilar to an audio-visual feed by another means, but could also be argued that the code is running locally and generation of the audio-visuals is unique to that computer. For example, if any random number generation is used, then each performer/observer is generating their own unique random number stream. Each performer/observers experience will be different and almost exactly the same.

[3] ACE Editor - https://ace.c9.io Firebase - https://firebase.google.com Firepad - https://https://firepad.io

[4] Extramuros - https://github.com/d0kt0r0/extramuros Overtone - http://overtone.github.io LuaAV - http://lua-av.mat.ucsb.edu Extempore - http://extempore.moso.com.au Gibber - http://gibber.cc

[5] Example live-coding performance where neither performer was present. Performed at Abrons Arts Center as part of the New York City Electronic Music Festival, 2017. https://vimeo.com/224842457/35e2eeeb0f

3 Collaboration at a Distance

The following are reflections on what occurred when the collaboration started working at a distance during the development of The Dark Side and what happens when collaborators have access to each others domains of expertise.

The following link documents many of those sessions. https://vimeo.com/album/4688973

3.1 Shawn's perspective

All things, even collaborations, encounter change. In our case, the physical displacement created an artistic challenge which we decided was worth embracing. While the initial goals of the software didn't reach their mark, I think we came across some new territory we initially didn't intend, that being how your collaborator (a likely worse than novice in your field) can change your code.

What did emerge after weekly telematic rehearsals was me trying to see if Ryan would notice changes to that sound that I would make. In simple scenarios I would change samples and see if he noticed. In more complex scenarios, I would hide some Tidal code way at the bottom of the buffer and wait for him to notice. Occasionally I would run the code and then delete it, so as to hide the evidence. Often I would cause errors which gave me away pretty quickly

After doing this, Ryan would start changing my code, which, personally, I feel like had more drastic affects. Frequently these changes were incredibly difficult to track down and work with. Watching some of the recording, I think Ryan was much more successful and adopting my changes to the audio than I was with adopting his changes to the visual.

My impression from this experimentation was that my limited audio attempts were just enough spark to kick-off another cycle of exploration when the current rehearsal run had become stale.

3.2 Ryan's Perspective

Like many collaborative processes the development of The Rebel Scum is more or less typical, especially regarding the desire for increasingly responsive control and meaningful reactivity. This becomes challenging with the similarly typical physical displacement of collaborators from one another. The Dark Side of The Force is still not at the point we would like it to be, but the transmission of textual changes (client to server to client) as opposed to streaming audio and/or video not only preserved the fidelity of our respective contributions, but enabled the type of participation that leads toward our next planned version of The Dark Side.

The integration of both languages into a shared text buffer has led to interesting extrapolations from our previous aesthetic directions by virtue of our collective desire to challenge each other through multi-lingual sabotage. The idea that I would need to go back and "fix" a change made by Shawn is coupled with a desire to see the change as an introduction to a new musical direction. Approaching these acts of sabotage as creative challenges has led to intriguing musical directions that I would not have otherwise anticipated, and further removes our performance practice from the formality of *Kessel Run* and *Sarlacc*.

4 Conclusion

The current state of The Dark Side has allowed the authors to continue their ongoing collaboration by facilitating a fast and reliable method for sharing textual information in a shared buffer experience within a browser while rendering all content at high-fidelity locally. It is our hope that this new rehearsal method will lead to performance opportunities that may have otherwise been inaccessible for practical reasons, and at the very least allow us to continue developing our work.

4.1 Acknowledgments

Thank you to many in the live coding community for continuing to open source their works. The Dark Side may not have been possible without the generosity and availability of others' prior work.

References

Lee, Sang Won, and Georg Essl. 2015a. “Web-Based Temporal Typography for Musical Expression and Performance.” In *NIME’15 Proceedings*. New Interfaces for Musical Expression.

———. 2015b. “Live Writing: Asynchronous Playback of Live Coding and Writing.” In *ICLC ’15 Proceedings*, 74–82. International Conference on Live Coding.

Shawn Lawson, Frank Appio, Ryan Ross Smith. 2016. *Closing the Circuit: Live Coding the Modular Synth*. International Conference on Live Coding.