To Stream the Impossible Stream

Will Coeur, Iowa State University

The year is 2020. The world is facing an unprecedented global pandemic unlike anything seen by any living generation. Industries around the world are having to adapt to near insurmountable challenges to define a safe, new "normal." Some industries, such as Theatre, are facing the harsh reality that adaptability may not be logistically possible, and are therefore shutting down, hoping such actions will prove to be merely a hiatus and not a final chapter in their stories. The question of adapt or hiatus is, for most, a question of money and resources. For some, however, it is a question of experience. Is it possible to produce a quality product worth the investment given the resources available and the obstacles to overcome? For the Theatre Industry, this question of "quality of experience" weighs heavily on the institutions of higher education where many of the next generation are just starting their journeys. In an industry so anchored in learning by doing, is the act of discussing theatre and techniques with one another through a computer screen enough to properly equip the next generation of theatre artists with the skills and knowledge needed to succeed, or is the quality of the experience too burdened by a lack of practical application to properly educate the current student body?

In the height of the COVID-19 pandemic, the Iowa State University Department of Music and Theatre faced the difficult challenge of providing strong, practical learning opportunities while simultaneously attempting to mitigate exposure and possible illness. The first step to meeting these challenges was to remove live audience from the equation. Our main theater, Fisher Theater, is a standalone building that is almost exclusively used by the Department of Music and Theatre. With the removal of live audiences, our environment became much more manageable in terms of risk factors. Further steps to mitigate risk around our productions included socially distant rehearsals and the requested use of face masks at all times while within Fisher Theater, among other policies. With a "new normal" in place, we shifted our focus towards creating any kind of experience we could for our students. The removal of live audiences meant that the only consistent way our productions could reach the public was through live streaming and pre-recorded broadcasts of our performances. Like many academic programs and professional companies across the world, we were forced to dive into the deep end of the stream, as it were, as we were ill-equipped to take on such an endeavor. And thus began my own year-long journey towards creating that practical "quality of experience" our students were so readily hopeful to achieve.

The obvious first hurdle was a complete lack of equipment to support such an endeavor. Up to this point, our department had always rented microphones only for our musicals once or twice a year. Now facing, at minimum, a full season of shows that would require a way of capturing the performers' voices, it made much more sense to purchase. Using a USB Audio Interface, this was arguably the simplest part of the puzzle as the various pros and cons of many wireless microphones in a theatrical setting are heavily documented across a variety of accessible platforms. A fairly self-explanatory audio purchase later, I was ready to begin the deep dive into the world of video.

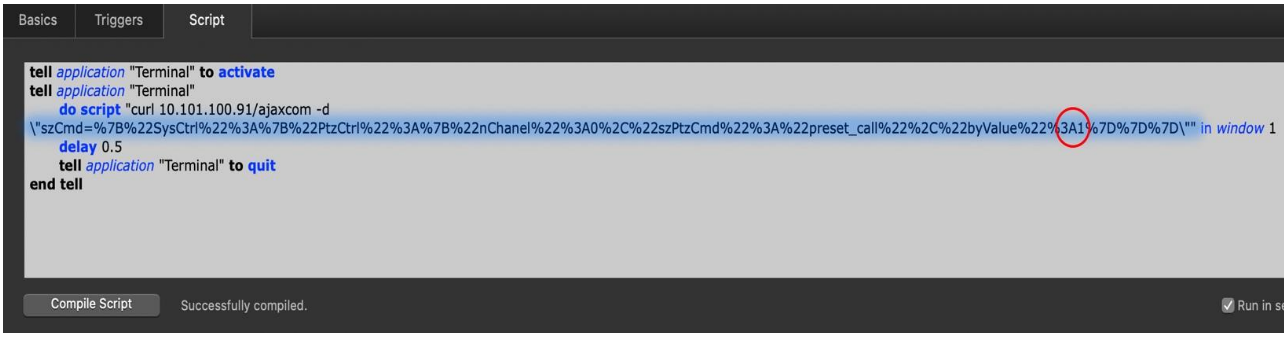
The challenge with video in Fisher Theater is rooted in the building's utter lack of performance infrastructure. It is a classic mid-70's theater where the only thing that connects the booth in the back of the house to the stage is the dimmer rack and a small amount of XLR panels. All other networking and media connections such as ethernet or HDMI must be done through long, individual runs of cable that can span anywhere from 200'-300'. While this poses a logistical obstacle, the real challenge is figuring out the right equipment needed for the project. One wrong choice can not only cost more money, but the time needed to return one item, purchase another option, and then wait for that new equipment to arrive is often not available even in a more spread-out Pandemic Theatre schedule. The fiscal equivalent of "measure twice, cut once." Fortunately, with a nation-wide shutdown in March of 2020 and no summer schedule to have to adapt around, this gave me approximately four months to sit down, do the research, demo some products, and do it right.

The first step was identifying how we wanted to integrate video into our process. Given Fisher Theater's existing infrastructure, but not knowing exactly how we might want to use this new element, we needed a system that could be very flexible and operate over a long range. While I explored a variety of options, I ultimately decided to focus my research on HDBaseT-capable technology, as I had already invested in that technology in the realm of projections pre-pandemic. HDBaseT is a long-range multimedia distribution technology designed to work over a single ethernet cable (Category 6a or above, recommended). The beauty of the HDBaseT technology is that it not only lets you transmit and receive various kinds of data such audio, video, and network data, but it also allows you to power certain devices, such as cameras, over ethernet up to 328'. This versatility, combined with the fact that I had previously purchased other HDBaseT components, made this the correct choice for our needs.

Now that I had narrowed down the integration technology, the next step was to find cameras that both utilized that technology and could meet our fundamental goals. The name of the game was versatility, so I immediately leaped into the world of Pan/Tilt/Zoom cameras. The thing about PTZ cameras is that many of them are designed to work with their own prebuilt system such as an app or a small controller often with a joystick and some other fine motor controls. From a purely plug-and-play standpoint, this is fantastic, simply feed the company's prebuilt controller into your system and you are good to go. However, it was a personal goal of mine to ensure that whatever we did still felt like theatre and not like broadcast television. So, I took a page out of the Prop Designer handbook and decided to take something meant for a specific kind of task and repurpose it to fit within my own theatrical construct even though it was likely never intended to be used that way. What this meant to me was finding a PTZ camera that utilized the HDBaseT technology, could capture and transmit quality video over potentially long distances, and, finally, could somehow be controlled by the Figure 53 software, *QLab*. That final requirement led to a several-week rabbit hole of research focusing on what combination of AppleScript, terminal commands, and sorcery could be implemented to control everything via a simple GO command. After a couple of weeks of searching the internet, and more than one instance of literally clicking in circles through the same few forum posts, I was able to find a potential answer: *cURL*.

The project known as *cURL* is a free and open command line tool (“curl”) and library (“libcurl”). For our purposes, I focused solely on the “curl” command line tool. Through the use of “curl” in the Command Terminal app on Mac OS, I realized I could send any URL, or web address, to the specific IP address of any technology that resided on my closed network. When we are on a website such as Google, any button we click that sends us to another page, opens another pop up window, or even just highlighting some text does so by making a specific network request. These network requests can be monitored using the Network tab in the Developer Tools of your preferred web browser. I utilized the *Google Chrome* web browser to monitor my network activity. Many devices, such as cameras and projectors, that have networking capabilities are often able to be controlled through a web browser interface accessed with that device’s specific IP address. My theory was that, if I could monitor the network activity of these browser-based interfaces, I could identify and match specific activity with specific commands. From there, I could utilize “curl” to mimic that specific network activity via the Command Terminal being opened and run by AppleScript through *QLab*, thus recreating the behavior of the browser-based control in a sequential cue list programmed within *QLab*.

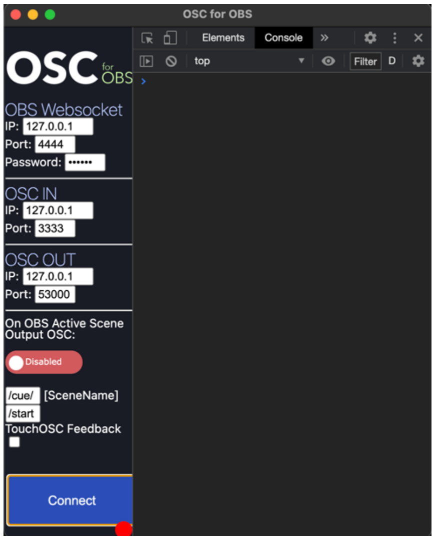
Following the mantra of “test twice, buy once,” I wanted to confirm my hypothesis before putting the department’s money where my mouth was. Using my HDBaseT Projectors as test subjects, I proceeded to log into their IP addresses via my web browser. With some trial-by-fire testing and some additional research, I was able to discover “curl” commands that would allow me to mimic the network activity I was seeing within the devices. In this case, I was able to create an AppleScript within *QLab* that could turn the projectors on and off. Following this success, I took steps towards making a final choice for our future PTZ cameras. I partnered with a local A/V company to demo the Datavideo PTC-140T camera. This particular piece of equipment has the ability to store up to ten presets using the included remote control. The concept of recordable presets made it an ideal candidate especially with multiple cameras each using several presets. Logging into the camera’s IP address, I was able to quickly identify the network activity related to recording and recalling presets. The key was identifying differences between the various preset activity which required looking at what *Google Chrome* labels the “Payload.” The Payload allows for modifiers of similar network activity such as Preset 1 versus Preset 2. An example of the “Payload” is highlighted in the image below with the modifier circled:



```
tell application "Terminal" to activate
tell application "Terminal"
	do script "curl 10.101.100.91/ajaxcom -d
\"szCmd=%7B%22SysCtrl%22%3A%7B%22PtzCtrl%22%3A%7B%22nChanel%22%3A0%2C%22szPtzCmd%22%3A%22preset_call%22%2C%22byValue%22%3A1%7D%7D%7D\"" in window 1
	delay 0.5
	tell application "Terminal" to quit
end tell
```

The value that follows the “3A” term in this Payload for recalling a camera presest indicates which preset number the camera is trying to recall, which I realized by comparing the Payloads of multiple camera preset recall commands, as this was the only value that changed consistently. With the proper modifiers identified, it was now possible to test the limitations of the camera. As previously stated, the Datavideo PTC-140T camera can store up to 10 presets via the included remote control. A quick manipulation of the Payload in an AppleScript revealed that the camera's software could handle any numeral in this value ranging from 0.1 to 99999.99999. Suddenly, my potential camera's recordable presets went from a maximum of 10 to a near infinite amount. This, combined with my AppleScript actually working with the equipment, solidified this camera as my final choice for our new system.

Moving through the season, we began to integrate this technology into our productions, starting off small with a simple single-position, two-camera setup utilizing wide shots and close ups, but with the intention of scaling up to a multi-camera, multi-angle system for our end of the year musical, *Songs for a New World*. With the two-camera setup, integration was as simple as plugging the connecting the cameras to our main video computer via the Datavideo HBT-11 Receiver and Datavideo Cap-2 HDMI to USB 3.0 Capture Box as there were more than enough USB ports to support the system. We now had all the hardware we needed for our initial setup, but now we needed a way to get our production out to the public. To meet this challenge, we turned to the MTI streaming service, ShowTix4U combined with the popular streaming software, *OBS*. *OBS* is a free and opensource video recording and live-streaming software solution that is the popular choice among live stream gamers and YouTubers. The software allows for a multitude of video and audio inputs and outputs, among other things, to be laid out and layered on each other into a single video output that can either be recorded, live-streamed, or both simultaneously. *OBS* allowed us the ability to input each camera feed individually and then crossfade between them seamlessly via Scenes. However, this added another chain to the link of things that needed to be controlled. Continuing with the same goal of having all elements controlled by a single GO through *QLab*, I now needed to find a way to control *OBS* via *QLab*. Fortunately, I was not the only one that had a need of that kind of control and another opensource project, *OSC for OBS*, had already been conceived. While still in its infancy, this project had made great strides in creating OSC (Open Sound Control) commands that could control *OBS* using a Node.js runtime environment as a user interface, pictured here:



OBS Websocket
IP: 127.0.0.1
Port: 4444
Password: ••••••

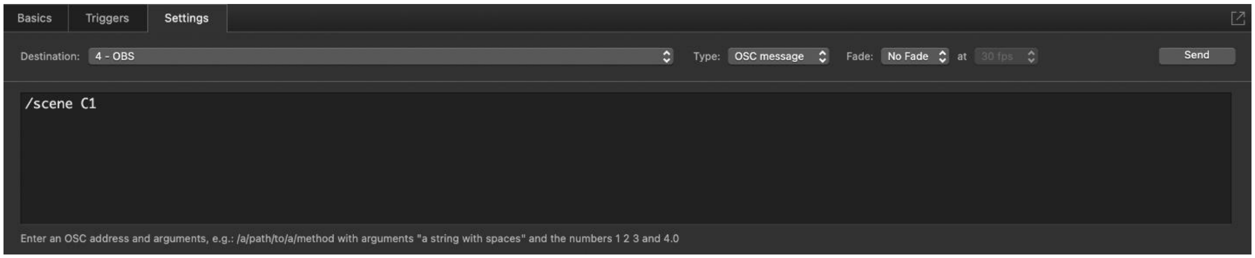
OSC IN
IP: 127.0.0.1
Port: 3333

OSC OUT
IP: 127.0.0.1
Port: 53000

On OBS Active Scene Output OSC:
Disabled

/cue/ [SceneName]
/start
TouchOSC Feedback
□

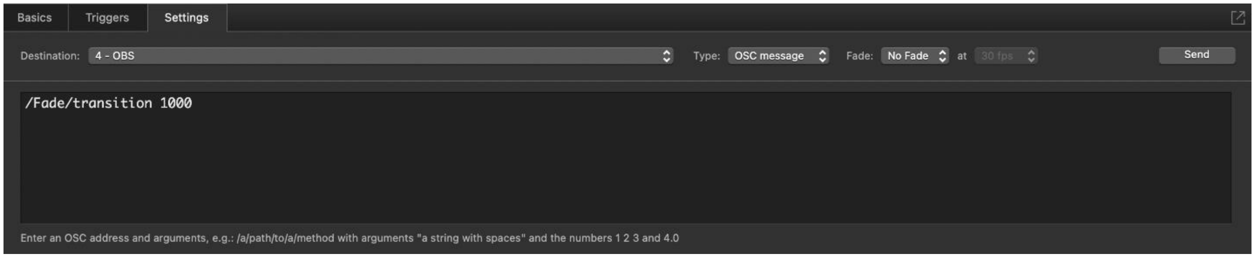
Using OSC Network cues in *QLab*, I could not only crossfade between various Scenes in OBS, but also adjust the transition times live. Examples of these OSC Commands are pictured below:



Destination: 4 - OBS Type: OSC message Fade: No Fade at 30 fps

```
/scene C1
```

Enter an OSC address and arguments, e.g.: /a/path/to/a/method with arguments "a string with spaces" and the numbers 1 2 3 and 4.0



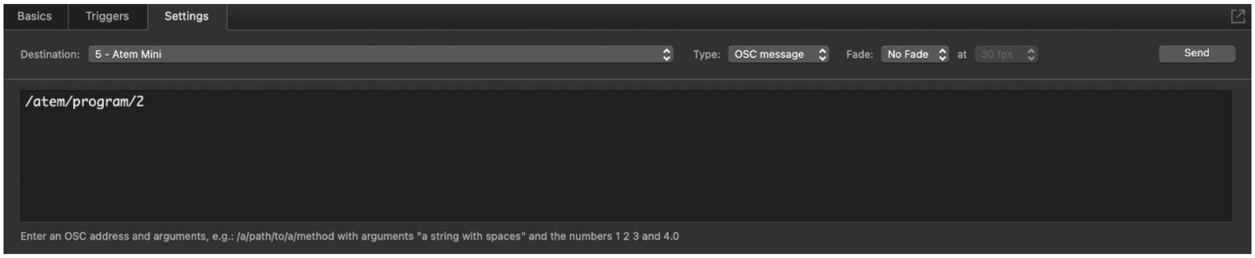
Destination: 4 - OBS Type: OSC message Fade: No Fade at 30 fps

```
/Fade/transition 1000
```

Enter an OSC address and arguments, e.g.: /a/path/to/a/method with arguments "a string with spaces" and the numbers 1 2 3 and 4.0

Now that we had extensive control of how our production looked, it was time to send our output to the public. ShowTix4U proved to be the most convenient service for this task as it was able to almost seamlessly integrate with the *OBS* software. Once we had created our production events through the ShowTix4U service, we simply had to copy and paste a streaming key from ShowTix4U into *OBS* and begin the broadcast.

With a fully operational live-stream system, the question was how we expand the system to accommodate even more cameras/technology without overtaxing our equipment. We had decided for the upcoming musical that we wanted at least four cameras from varying angles as well as possibly adding in a handheld camera into the system. We quickly concluded that a wired over-the-shoulder camera was not fiscally possible. Instead, we chose to leap into the less guaranteed world of wireless capability. While a riskier option, it was agreed that the cost-effectiveness and ease of a wireless option were positives that far outweighed the negatives. To accomplish this task, we turned to some pre-existing hardware in our inventory. Our final solution ended up being an old iPhone on a self-stabilizing selfie stick AirPlaying to an Apple TV that we connected to the computer. With all cameras ready to be integrated, we just needed to figure out how to connect everything to the computer. In our simple, two-camera system, we had plenty of ports for connecting our cameras with no issue using USB 3.0. This was no longer the case, as our number of inputs far exceeded our computer's capabilities. Often, the solution to such an issue is to add more computers, but we needed everything to land in a central broadcasting station and I wanted to add as few failure points as possible so that was not a viable option. The time had finally come to step into the world of video mixers and switches. As this part of the system ideally also needed to be controlled by *QLab*, my search quickly narrowed to the Blackmagic line of video mixers, finally deciding on the ATEM Mini. The ATEM Mini allowed us to feed up to four different video inputs into a single video output that could then be connected to the computer and controlled by OSC via the network proxy program *atemOSC*. This network proxy allowed our control computer to communicate with the ATEM Mini directly over the network and could activate certain functions, such as camera switching, using its simple, preprogrammed OSC commands in a Network cue in *QLab* such as the one pictured below:



Destination: 5 - Atem Mini   Type: OSC message   Fade: No Fade at 30 fps

```
/atem/program/2
```

Enter an OSC address and arguments, e.g.: /a/path/to/a/method with arguments "a string with spaces" and the numbers 1 2 3 and 4.0

After some creative camera mounting and more than a few long cable runs we were finally ready to start programming the show. While the design and use of the system had been almost solely my task up to this point, as Lighting Designer for *Songs for a New World*, I needed someone else to step in and take on the monumental task of designing the camera shots for our production. Senior Performing Arts major, Calvin Clark, stepped up and became the production's cinematographer. With a bit of coaching on how to set up, record, and recall presets with our new cameras, Mr. Clark jumped in with an enthusiasm that would make any educator proud, designing more than 25 different static camera shots across four cameras as well as conceptualizing and executing a variety of moving handheld camera shots throughout the performances.

So, that was it. The system was designed and purchased, the sequence was storyboarded and programmed, and the show could broadcast to the public. Now I could focus on my lighting design, or so I thought. *Songs for a New World* would be the first live-streamed play out of Fisher Theater. All other Fisher Theater productions up to that point in the season had been pre-recorded. After recording the production, but before releasing to the public, we would send our recordings to Rev to have them closed-captioned to add a level of accessibility not always found at Fisher Theater. Having set that standard for our "streaming" season, our students insisted we at least investigate if it would be possible to do the same for *Songs for a New World*. While I knew it would not be the easiest task, it was not a request I wanted to deny. With a bit of thought and a massive amount of assistance from the production's Assistant Stage Manager, Stephanie McClelland, we were able to transcribe the script into a series of screen-width Text cues in *QLab* that could overlay on top of the live-stream. However, knowing the pacing would never be consistent every night, I needed a way to fire only those Text cues while the Stage Manager ran the camera Script cues. This led to the addition of another computer running a different instance of *QLab* firing a singular AppleScript on repeat that would fire the next cue in the "Closed Captioning" Cue List in the main *QLab* file run by a stage crew member with script in hand.

And the show went on. More than 200 cues of live camera switching across the 25+ designed camera shots, integrating live microphones and band through an audio interface, and over 1000 closed caption Text cues just so we could share that typical "quality of experience" with our students and audience that we had all somewhat taken for granted prior to the pandemic. There were stumbles along the way, but the final product is something I am so proud of, not just for my own contributions, but for what we, as a department, came together and created.

Works Cited

Stein, Liad. “About Technology - HDBaseT.” HDBaseT, 1 Aug. 2022, hdbaset.org/about-technology.

Curl - Frequently Asked Questions. curl.se/docs/faq.html#What_is_cURL.