How I lost 150 lbs. thanks to Max/MSP!

Back To My Routes: Freely Improvising with Max/MSP

Jeff Kaiser, (admin-{at}-pfmentum.com)

Abstract: While conferences and symposia are replete with papers and presentations with the admirable goal of furthering the development of "software as composition," it is less common to encounter similar discussions of software-mediated improvisation in which the "composer" and "performer" are one and the same person. In designing software for improvising musicians, different needs must be addressed, which would include an individual's unique improvisational view, idiosyncrasies of his/her instrument, and flexibility within a variety of live performance situations (rather than repeatability within a proscenium or otherwise controlled environment). This paper describes why I moved from a hardware-based to a software-based music performance system, and why I chose Max/MSP as the most flexible system for meeting my personal needs as an improvising musician.

1 Losing the stomp box: How a patch cord paradigm saved me excess baggage charges and personal grief on planes, trains, and automobiles

So then, the question stands, "How does a non-academic, free jazz, improvising quartertone trumpet player with no computer language programming experience, one that has been using low fidelity guitar 'stomp boxes' on his horn since the mid-1990s, get into writing his own software?" At first, I was led to the manna of software for purely practical reasons: I was tired of carrying gear. However, what was originally a practical decision soon became an aesthetic one when I realized the advantages presented by the open-ended nature of a modular software program, namely, that this software would enable me to go beyond the limitations of the pre-existing singular functionality built into the individual hardware based audio processing devices (stomp boxes) I had been using for years. Even when those stomp boxes were physically connected together in multiples, the possibilities nowhere approached what could be done with a modular software program.

In the Beginning

There I am, in London, August 2005, one week after the terrorist attacks on the subways and busses, to do my first concerts abroad (unless you count that strange gig in Tijuana, Mexico a few miles south of the California border at the old paper warehouse with Eugene Chadbourne as "abroad"). My two gear bags weigh over 150 lbs. It is not just the stomp boxes, but also the power supplies, power converter, plug strips, small mixer, expression pedals, microphone, and all the cables to connect them. Add to all that my trumpet, flugelhorn, mutes, and clothing. I haul all the stuff through Heathrow from the airplane luggage retrieval area through customs, where I am deftly and thoroughly questioned and inspected, all the while being eyed by machine-gun carrying gentlemen. I load the gear onto the train and, in a comic moment worthy of a Hollywood B movie, I delay the departure as I get stuck in the doorway with my Pelican-style cases preventing closure of the sliding doors. For my reward I receive that look of condescension that can only be delivered by natives of the British Isles. I get off the train, and into a cab, in which I barely fit with all my equipment. It is quite a chore to do that on any day, but with the remnant physical memories of the cocktails imbibed on the flight, which co-mingle with the leftover effects of the sleeping pill prescribed to ease the overseas transition, it becomes an absurdist play with my gear cases turning into fascist rhinoceroses of independent mind and will. That is my life for the next week: hauling ridiculous amounts of gear from hotel to cab to club and back again.

All the time my band mate is carrying only a laptop bag and a horn case for his gear.

We then come to the climactic gig of the week as opening act for legendary saxophonist (and one of my personal musical heroes) Evan Parker, at which point my "high end" ring modulator dies at sound check. This forces me to re-arrange my whole set-up, creating problems with the other

stomp boxes in gain staging and throwing me off for the whole set. That is it. Never again. Pain and discomfort are great motivators. I swear to never again travel with all that hardware and, instead, to find a solid, reliable laptop-based solution for my love of processing my trumpet.

Little did I know, not only would I come up with a personal solution to this situation (one achieved by many other improvisers before me, notably, and to impressive results, George Lewis [Lewis 2000, 33-39] and Evan Parker) but I would also enter into a world that would provide me with an even greater variety of timbral and spatial possibilities, reconnecting me with the reason I first went into electronically processing my trumpet.

2 Creating a software instrument to be played, versus software that is a performance

I am speaking here, first and foremost, about my musical perspective as an improviser. I am a musician, and I want to make music, not create the latest or greatest theoretical software construct, but a practical instrument to perform with at nightclubs, galleries, and concert halls. The idea is that the technology would be at the service of artistic vision, creating a long-term relationship with the music making—this as opposed to the short-term relationship created when momentarily inspired by the novelty of an effect. This brief and torrid love affair with novelty is most apparent in my past need (and others) to buy the latest greatest box that does one really neat thing to your sound. (Did I really need that $350 step-sequencing ring modulator?) And then it gets used to death, after which the joy of the original novelty wears off and you're left with a heavy, over-used piece of obsolete processing hardware with its boutique power supply that won't work on anything else and a chunk of money gone from your banking account.

No, I am not bitter.

So, in choosing a software-based solution and the development of the software instrument, considerations of the long-term artistic/musical vision would also necessarily go hand in hand with directly addressing the practical needs of an improviser:

1) Variety of sound processing options

2) Immediacy of operation

3) Dynamic control (i.e. the ability to "play," with ease, the parameters of the given sound processing options)

My goal, with the above three elements in mind, was to create a single, personal software instrument that would simply and fluidly interface with the trumpet in a variety of performance environments. It would become part of the instrument, not just an added effect, and it would be as seamless as possible in its interactivity in an improvisational musical setting, within the confines of the physical restrictions imposed by my playing the trumpet.

The restrictions/constraints present in the design and control of any software instrument used by an instrumentalist (due to the fact that hands are being used to play said instrument) must be addressed. Evan Parker has solved that problem by delegating other performers to specifically handle the processing of the acoustic instruments [Feller 2001, 80-82]. Another option would be to automate the processing of audio, as woodwind multi-instrumentalist Andrew Pask of Cycling 74 has done with his personal software system [Pask 2005]. I also did try a few optical/video-based controllers, but deemed them too unreliable in crowded nightclub situations, but I am certain this will change with time. As a trumpet player, a small advantage (over woodwinds and trombones) for operating the processing equipment is that you do have one hand free and like the other instrumentalists, if seated, the use of both feet. So, in my desire to keep choice and some control over the decision making of the audio processing, and a desire to play the software as an instrument, I decided to go a route I was familiar with, that is, to use pedals and buttons, a setup that would be solid, reliable, and simple to use. Not to mention an easy performance transition from my past use of stomp boxes.

Practicalities would also include considerations of cost, availability, and weight. Cost and availability are factors because on the road things break and need replacing, so redundant devices would need to be carried or available at any local music store. (This would avoid

situations such as what happened opening for Evan Parker in London.) I consider weight an important factor, not only because of excess baggage charges, but also because I simply can't afford a roadie and it is not convenient, or desirable, to carry large amounts of hardware.

3 Software: Why did I choose Max/MSP?

In the past, a combination of preconceptions and fear of programming had kept me from entering into the software world. I had fallen into tacit agreement with the negative perception of modular software programs such as Max/MSP and Pd, a perception that exists among some of the less academically oriented improvising musicians using electronic audio processing. This perception is that languages such as Max are for software based compositions, pure academic/analytical work and meant only for the pocket protector and slide rule class of musicians, not for working-class players on the street. Working and discussing with improvisers, DJs, artists working in electronica, looping, noise, and other forms involving the mediation of electronics, I heard it over and over again, "Do you want to program? Or do you want to make music?" "Software as composition" is a fine and admirable model, but, as I have learned, it is only one possibility of an incredibly versatile technology that is musically, in and of itself, genre and intention free.

As I began searching, I immediately excluded the more complex languages such as C++, an obvious choice if I were a software engineer. However, as a musician in the working world, I needed something more readily accessible for my personal skill set, which includes some technological background, but no real programming experience.

Limiting myself to the computer I owned, a PowerBook laptop, I examined and considered programs such as Ableton Live, Reaktor, and various VST hosts. But I wanted something that would allow me to create an even more unique, idiosyncratic system that would be built specifically around my personal needs and artistic vision as a composer/performer.

Because I used multiple hardware looping pedals, Radial was a possible choice as well, with its looping features. But when I discovered that Radial was written in Max/MSP, it pointed me in the direction of modular software programs. Along the way, I also looked at Pd, a powerful, open source modular program, but ended up choosing Max/MSP. The reason I chose Max/MSP was because of my existing ties to users of the application and also an active online discussion group that I was introduced to early on. (On its surface, the discussion group seems to consist mostly of curmudgeons, skeptics, cynics, a couple of drunks in a basement in Düsseldorf, and one nice guy named Stefan with a multitude of consonants for a last name that are currently indecipherable by the linguistic limitations of this writer. But the group is a great resource, and there are some helpful and creative folks there.)

In the beginning, keeping it real basic, I used Max/MSP as a fully customizable VST plug-in host. That was a great point of entry into live processing for me, and I've used it to successfully introduce others to the joy of software based sound processing. It gets past the whole, "Do you want to program or make music" argument. VST plug-ins provide the immediate musical gratification that is part of the allure of stomp boxes, and it requires little programming skills to set Max/MSP up to do this. In fact, with so many pre-existing plug-ins freely and readily available on the Internet, you can have a basic VST host up and running in an afternoon.

So that is why I was first attracted to Max/MSP. The second reason was that it would, in the long run, provide a way to create a unique voice rather than a pre-fabricated electronic one following someone else's vision and design. A simple VST host would not satisfy that desire. With Max, as I learned, there is a great ease in creating unique audio processing patches on a basic level, and the ability to also create audio patches of increasing complexity as your skills improve organically, at a natural and individual pace.

The first priority listed earlier is to have a variety of signal processing options. The amount of variety achievable by software is, of course, vast. If you have, say, six stomp boxes, there is a possible 720 different ways to arrange/order

those boxes. (This is just a basic reordering of the boxes; the number does not take into account the parameters available on each box.) But the physical patching required to reorder these on-the-fly at a gig would be incredibly time consuming, taking away from the immediacy of improvisation. So with boxes, you end up with static ordering, limiting the timbral variety readily available at a gig. To increase variety with hardware, you would, obviously, have to add boxes. Doing this adds variety but increases the schleppage factor, i.e. increasing the amount of gear you travel with, cables, power supplies and the time spent plugging it all in. In my current software instrument, I have twenty-seven processing modules. If Google calculator is correct, that gives a possibility of $1.08888695 \times 10^{28}$ possible ways to arrange/order the modules. (A wee bit more than 720, and a truly unfathomable number for a trumpet player such as myself, hence the use of Google calculator.) Using language as an analogy, the numerical difference is an incredible leap in readily available vocabulary. This variety is all possible with the push of a button in software. In truth, this was too vast for me. I've kept the modules in serial order in my rig. But even with this pre-determined number and order of modules, combinations are frequently, and pleasantly surprising. This mix of prepared/known combinations of patches and surprise combinations keeps one fully engaged with the instrument. But the big selling point for me is the ability to realize original processing ideas. This is where other programs fall short, as their "building blocks" are too big. The smaller building blocks of the modular software program enable me to create new components in a fairly straightforward, almost intuitive way (depending on the complexity), and deviate from the path of being reliant on VSTs developed for other people's artistic needs.

So, I ended up with my current rig, a hybrid of modules housing a variety of VSTs (that both emulate former stomp boxes and sophisticated, high-end, rack-mounted gear) and including original audio processing patches that are part of my expanding vision for performances. Over the evolutionary period of the development of my instrument, the interface itself has become more "set," developing at a slower rate, while my development of original processing modules has picked up. At the current pace of development I will probably abandon VSTs entirely in the next year.

4 Interface: The intersection of artistic vision, musical goals, and ease of interaction

The second priority is immediacy of operation. The oft quoted, "In fifteen seconds the difference between composition and improvisation is that in composition you have all the time you want to decide what to say in fifteen seconds, while in improvisation you have fifteen seconds," attributed to Steve Lacy by Frederic Rzewski [Bailey 1993, 141] is clever, but the truth is you might not even have fifteen seconds. Improvisation is about immediacy. Whether developing an improvised solo or improvising in a group setting, you need to be able to musically react to your individual thought or group idea quickly, intuitively, bordering on the instant. Any software instrument, interface, and controller used in improvisation needs to accommodate this immediacy. In light of this, the following becomes self-evident: turning things on and off and dynamically changing (playing) the parameters of the given module cannot be accomplished with a mouse, track pad, or track ball. No mouse. I'm a horn player. It is too difficult to play and mouse at the same time. So that means I need to get buttons and pedals working. This is where Max and other modular programs surpass the pre-made VST hosts I've looked at. All software allows for pedals and buttons. But in my goal of light, portable, and reasonably priced interfaces, Max allowed me to save a lot of weight and cost by bypassing midi interfaces with my controller pedals. (See the appendix for more on my use of controller pedals.)

Other interface and hardware control considerations: They all must be portable, practical, reliable, rugged, replaceable, and reasonably priced.

There is always a "better" sounding interface than whatever you chose. There are always controllers having more features than whatever you chose. And there is always a techie only an email list away that is going to tell you what you

really should have bought. But are the "better" hardware devices right for the road?

1) Any hardware chosen for the road must be eminently portable. Maybe if you only do one or two gigs a year, you might not mind carrying large pieces of gear. But with a regular performance schedule, portability becomes a necessity.

2) It must be practical for the club environment. Stages, if there are stages, at nightclubs tend to be small and tight. In this way, clubs almost demand a de-evolution of some equipment, such as using less pristine microphones and interfaces. Whereas one microphone might get a quantifiably better sound in the studio, it might be bad onstage as it picks up every other instrument and easily causes feedback with stage monitors. Not to mention its fragility and replacement cost. And high-end pre-amps and interfaces are mostly relatively big, bulky and fragile. So this point also cuts out the exotic. Occam's Razor applies to the selection of hardware. After trying out and drooling over high-end equipment, I ended up choosing simple, common controllers. I am not the poster child for conspicuous consumption in the musical gear arena. There is a "latest and greatest" mentality surrounding audio gear and interfaces, but when you throw in the idea of practicality in a club arena, the expensive stuff, perfect for installations and performances in art galleries and concert halls, becomes a liability. Using the Lemur at a bar? I freak out when someone puts a wine glass near my Oxygen 8, but I've seen a milling audience member go to set a frosted beer mug on a friend's Lemur. (The fan obviously mistook it for one of the other twelve-inch LCD touch screen beer coasters lying about.) Along this line, I also give a vote for USB or Firewire powered devices: I once emptied the spit-valve of my trumpet on a band mate's power supply which lay unnoticed at my feet on a small stage. The "liquid" short-circuited his interface causing his computer to crash mid-performance—an event now known as The Spit Incident of '06.

3) Reliable. Goes without saying. Go too cheap and you'll find your gear is always breaking down or has poorly written drivers causing crashes.

4) Rugged. (See above mentioned "Beer on Lemur" anecdote.) Also, Jeff's Axiom: When playing nightclubs, all gear placed on stands, tables or raised supports of any kind will eventually succumb to the combination of gravity and a drunk fan. Which brings us to the last required points for the gear:

5) Replaceable and reasonably priced. It must be easy to replace. If you land in Boise, Idaho for a gig that night and check your gear and find your Lemur is not working, you better have a backup. Which can be pricey. Whereas, if your controller is something common and functional (but certainly less hip) such as an Oxygen 8, you can find a replacement at almost any local music store.

5 Future directions and conclusions

I am new to this, and have made many errors along the way of transitioning from hardware-based boxes to software. (I would also like to take this moment to publicly apologize for the possible permanent damage I did to the ears of the audience in my first laptop-based gig in Southern California at the beginning of 2006.) But I've learned this: the beauty of the software journey is that it never ends. As you grow as an individual musician and gain familiarity with the program, your software instrument is able to grow with you. And the big plus: you never end up with a closet full of stomp boxes morosely waiting and piling up for the sad, but inevitable, posting on eBay.

My advice for other musicians with no software programming experience wishing to go down the same path is: Don't be overwhelmed. Enjoy the exploration of options in software packages. In the end, choose a package that not only handles your immediate needs, but one that can grow with you into the future. This is why Max was a good choice for me, as Max can be as simple or as complex as you want it to be.

And some advice for those of you who have a friend addicted to stomp boxes and wish to convene an Intervention on his/her behalf: Don't get all complicated on them right away. If the first thing you present to somebody interested in a software system is a complicated and involved patch, and they don't have an engineering background, I guarantee they will be intimidated

and put off. The presentation of information is far different than actual education. Help by giving them a simple and clear entry point, where inquisitiveness and the possibility of the fulfillment, even the stirring, of artistic vision can enable the education to begin. This is part of what kept me from moving to a software-based system for years: I was intimidated by being presented with so much information. But show a person unfamiliar with audio software an individual VST object patch, and the way to patch a group of those together, and all of a sudden it is simple and clear, just like plugging in stomp boxes. After using VSTs for a while, you begin to get ideas for your own patches.

This is the beautiful cyclical nature of art inspiring technology, and technology inspiring art. For me, the best part: technology allows and encourages me to move further down the path in the never-ending quest of fulfilling a personal artistic vision.

Appendix: Current interfaces and hardware controllers

With so many boutique high-end devices available, the current selection of gear I use might seem a bit contrarian and unevolved, but the equipment decisions were based purely on the pragmatic necessities set forth in this paper.

Microphone: Shure Beta 56A.
This is a decent sounding supercardioid microphone that excels in a live environment by providing minimum feedback and maximum isolation from other sound sources.

Audio interface: MOTU Ultralite.
Reasonably priced, decent pre-amps, small, light, and has many analog in and outs to support my particular use of pedals.

Expression Pedals: Roland EV-5.
Reasonably priced, readily available. I was originally carrying around a Doepfer Drehbank to hook three of these into. It weighed a lot and was really overkill with its 64 faders. In looking to replace it, there were many custom options, usually heavy and pricey, or the lightweight but expensive Midi Solutions interfaces (individual controller converter boxes). With my current usage of five pedals, Choir Boy Andrew Pask helped me come up with a software solution, and I found the way to interface it: a three dollar stereo "Y" breakout that attaches to the EV-5's TRS plug. One of the mono plugs then goes into an analog output of the interface, and the other into an analog input.

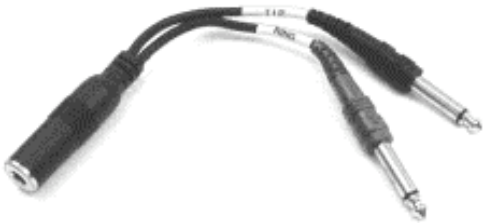

Fig. 1. Stereo "Y" breakout cable

This is light and inexpensive. No more big midi interface bricks to carry around.

I send a [cycle~] out of the audio interface into the EV-5 and then back into the audio interface where [peakamp~] measures any amplitude change introduced by the EV-5. This range of data can then be mapped into midi data (as in figure 2) or anything else you want it to become. This simple little patch has done more to save weight and hassle than any other single element in my rig.

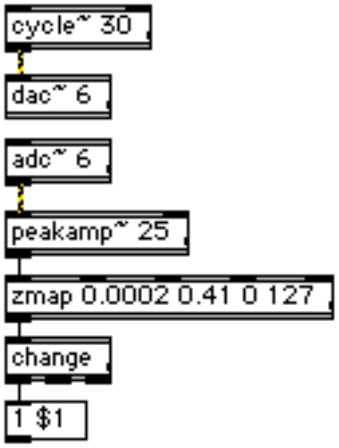

Fig. 2. Sine wave to data Max patch

Other controller: m-Audio Oxygen 8.
Keys provide on/off information and there are plenty of faders for my needs. Practical, inexpensive, and replacements are readily available.

References

[Bailey 1993] Bailey, D., (1993). *Improvisation: Its Nature and Practice in Music*. Da Capo Press.

[Feller 2001] Feller, R., (2001). “Evan Parker Electro-Acoustic Ensemble: Drawn Inward”, Computer Music Journal 25.2.

[Lewis 2000] Lewis, G., (2000). “Too Many Notes: Computers, Complexity and Culture in Voyager”, Leonardo Music Journal 10.

[Pask 2005] Pask, A., (2005). Personal conversations with the author.