Tom Erbe Interviewed by Theodore Gordon

TOM ERBE[1] and THEODORE GORDON[2]

[1]University of California San Diego, United States. Email: tre@ucsd.edu
[2]Baruch College, City University of New York, United States. Email: ted.gordon@gmail.com

In this interview, Tom Erbe reflects on his three decades as a developer of computer music software and hardware, both freeware and commercial. Tom met with Theodore Gordon on 8 December 2021 and discussed the beginnings of *SoundHack* and its roots in experimental music studio practice and hacker culture, ideas behind the design of sound processor interface and the shift from experimental software development to Eurorack modular hardware design.

Theodore Gordon: I think I had the most questions about the first and the third sections of your article. The first section talks about the origins of *SoundHack* in terms of the specific project you were working on – Robert Ashley's *Improvement* – and the technological and cultural landscapes in which you were working. And the third section is when you discuss the shift from plugins as standalone software to, for lack of a better term, 'modular synthesis'.

Tom Erbe: Well, to address the first part of your question, David Rosenboom and I were in the studio for maybe two months just experimenting with effects and with getting unique sonic landscapes for each character, so we were encouraged to really freely do things with whatever equipment we had – we had mostly live equipment.

TG: Yes, I was going to ask about this – in the article you write that this work was informed by the studio as instrument concept. Where did that concept come from for you – what kind of an instrument is a studio? Is it a musical instrument, a scientific instrument, an instrument for performance?

TE: Oh boy. I'm not prepared for this kind of question! [laughter]. I mean, 'studio as instrument' is a concept that people kick around so often – you really listen to any pop music in the 1960s and 1970s and you can hear the studio entering into the music. At times it's more forceful, at times not; even in very early Les Paul songs you can hear things that could only happen in the studio, like double speed guitar. These are the types of sounds that caught my ears early on in my experience of listening. So I was certainly aware of this general concept of 'studio as instrument' for most of my life. Once you get in the studio – in this case a multitrack analogue studio – if you have any interest in those sounds at all, you feel like playing with effects. And in that particular time [1989], effects are fairly well developed, and most of them had MIDI control.

It was at a point where things were just evolving beyond the modular synthesis of the 1970s, so many of the effects we used still had voltage control inputs as well as MIDI. And the nice thing about those early MIDI effects is that often a lot of parameters were mapped directly to MIDI inputs, so we were modulating a lot of things via computer-generated MIDI. This is commonplace now, but then it was just being tried out. And for *Improvement*, I was also trying to hang things onto the score as much as possible – following the pitch of the passacaglia with certain drones or resonances, and trying to follow the time signature with all of my delays.

TG: So you're in a multitrack tape studio, and you're working with source material on tape – and you have these outboard effects boxes you can patch in to individual tracks. How does this connect to working with computer audio processes such as cross synthesis or sound file convolution? It strikes me that that's a very different world compared to computer music and digital audio. How did you connect your experience with computer audio to the *Improvement* project?

TE: Well, I did work at the UCSD Computer Music studio in the mid-1980s. It was the period in which they were developing *Cmusic* and the phase vocoder, and the whole thing was distributed on computer tape. I was especially struck by the sound of phase vocoding and convolution – in 1985 I had never heard anything like that before. So, with convolution you soon realise 'well, this is a way to make reverb or to make any kind of space', so you'd play with instrument note clusters and put your voice through that, and find all these things that were not really possible at the time. To me, these processes seemed very close to musique concrète techniques in how they treated sound. So, when Bob [Ashley] told us that he wanted 'metaphorical spaces' for each scene, I immediately thought 'oh, convolution is what we need here!'

At the Center for Contemporary Music (CCM) at Mills, we had just got a Macintosh II computer – which could handle floating point calculations – but what it lacked was software that could do anything like convolution. The first thing I did was to port

doi:10.1017/S1355771822000206

Csound to the Macintosh Programmers Workbench, a really early programming environment. But I also wanted to get convolution working. In a sense, this work with computer music was a little disconnected from the *Improvement* project, but it really was the sound of the convolution that I was after – that's the connection. And convolution was really the sound that Bob was asking for, but I couldn't program it while doing 60 hours a week in the studio.

TG: It sounds like you were thinking in terms of function: convolution produces an interesting effect on any source sound, and it functions essentially in the same way as outboard gear in a studio. It's a box with input and output, and in between there's something that happens to the sound material – whether it's on magnetic tape or whether it's digital audio. To me, this sounds like 'black boxing' the technology and really thinking in terms of input and output, as opposed to thinking about this from an electrical engineering or computer science perspective. Rather than being motivated by technical possibility, it sounds like you were motivated by the effect that this technology had on sound.

TE: What motivated me was that I knew what the sound of convolution was, and I thought that it needed to be available to more people. At Mills, we had visitors all the time who used musique concrète techniques as part of their work. For example, when Alvin Curran came in, he was taking all this stuff off his sampler, putting it on multitrack, layering it, filtering it, slowing it down, etc. This was right when I was developing the phase vocoder and the earliest versions of *SoundHack*, so I was able to share this with various composers who were around – they used the early software and their work while they were at Mills.

TG: I was curious about what kind of resources one would need in the fall of 1989 to do what you did with *SoundHack* – what kind of computing equipment you needed, how much training you needed, what kind of understanding of acoustics, electrical engineering, computer science, etc.? I ask this because in the article you locate this work within the hacker culture of the late 1980s and early 1990s, and although this may be a retroactive generalisation, that culture is often marked with a kind of democratic ethos or imagination – anybody could do it, it wanted to be free, it wanted to be bottom-up instead of top-down. What did it take for you to be able to develop *SoundHack*?

TE: I bought a Macintosh IIsi for myself, which I think was maybe around $2,000, and I had to buy a compiler which was probably a couple hundred dollars, and that was the financial constraint. I do have a degree in computer science, so I have that background, but when I learned computer science, we were learning FORTRAN and Pascal. Things changed very quickly in the early 1980s, as far as languages as far as types of computers – I was no longer working on mainframes. In fact, the Macintosh computer first came out when I was working at UCSD, and many of the people there disregarded it – like, 'why would you want to work with such a low end machine? The screen is so small with such low resolution, how could you do anything with it?' There was definitely a general perception that it was not adequate for our work

But the hacker culture was about using whatever you had at hand. And at that point people were buying their own computers, whether it was an Amiga, Atari ST or Macintosh. For example, I had an Atari ST at one point, and I built a little sound output board for it because it didn't have a nice DAC for the output. I found that you could get blank circuit boards that would plug into the Atari's printer port and just make it yourself. And the best thing was that I could write my own software to do Karplus Strong or whatever I had read in the *Computer Music Journal* that month [laughter].

So we were basically trying to get things working via non-real-time software. This was also the time that *Csound* was being written, and Paul Lansky wrote a lot of software, and a lot of composers were generating their own software for their own purposes. At Mills there was HMSL, an object-oriented compositional system based on FORTH. And there was this idea that, well, we have these computers, but no one's going to provide interesting music software for them. It's just not going to happen. So if you want something that's interesting, you have to build it yourself. Nowadays people often say 'to get something interesting do something original' using a basic DAW as a kind of neutral environment. But back then, it was, 'you can't even use any of the software that's out there to do anything – it's just not going to make new music'.

So that was that aim of the hacker culture. And people in the studio – John Bischoff, Chris Brown, Larry Polansky, Paul DeMarinis – I could name drop all day long – they were making their own instruments. If they wanted something they would go out and build it, and write the software, too.

TG: It sounds like there was a feeling of if not freedom, maybe expansion or possibility about the affordances of a general purpose computer. There were inputs and outputs for audio, and between those, you could write your own software that drew from these very rich resources such as *CMJ* and nascent communities that would share among themselves.

TE: Definitely – and I soon found that my talent was developing software, so I started going with that, and I made *SoundHack*, which really was the first thing I wrote.

TG: In your article, you write: 'There were no design or research motivations to *SoundHack* at this point. The focus was simply sound, synthesis and music.' I was curious about this sentence – why did you feel the need to make this clear?

TE: It started with a very utilitarian need, and that was just to convert sound formats from one format to another. The reason I did that, though, maybe did have a little more purpose behind it. As I mentioned earlier, Phil Burk and I had ported *Csound* to the Macintosh at Mills, and it spat out IRCAM format files. One of the projects I undertook at Mills was to get the more academic computer music tools available at the studio; I thought that we definitely should be able to use *Csound*, *Cmusic*, phase vocoding. These were things which at that point you needed to go to UCSD or MIT to do. At Mills we were constrained in budget, and we couldn't afford gigantic computers. There also was a much larger emphasis on live electronic music – an aesthetic difference which I maybe didn't quite appreciate at the time. But that was probably a good thing, because I just said 'okay, we're going to bring this culture into that culture, and make these things possible. People will hear more sounds, more ways of making sound, and this will be good for music.'

TG: I'm struck by the way you put this – bringing one culture into another – because it highlights a big difference in musical and research cultures at these different institutions. IRCAM, MIT, and even UCSD – these are big budget research institutions, often with state-sponsored top-down research agendas; and on the other end you have Mills College, which saw itself as a kind of scrappy, motley crew of people who just figured things out for themselves – especially considering the CCM's origins at the San Francisco Tape Music Center.

TE: One of our favorrite past-times – one thing we would do constantly – was visit various electronic surplus shops around the Bay Area. And you'd scrounge for, like, old microcomputers, stuff that would allow you to do something interesting; or maybe you'd find some kind of 'ultrasonic detector' or something like that which you could use in your next piece. I remember many car trips just running from one surplus store to another.

TG: In a way, it sounds like what you did with *Csound* and the research from CARL was similar: you were looking for surplus software that wasn't being used in the ways you thought it could be used. And so you basically took something that was available, given your access to these institutions, and treated it like surplus.

TE: I definitely did have thoughts about access at that point; I really thought that these tools were no good if they were just held by a few people. And in many ways, the work of the people who *did* have access to those tools could become too much a part of the tool, too: those people shouldn't be able to have basically complete control over the tools and the sounds made with those tools just because they had institutional access. It was time to spread it around a little bit.

TG: What role do you see the software engineer having in the creative process of the musician who uses their tools? Do you think software should be transparent, precisely and accurately controlled by the user, or maybe that some user control should be sacrificed in order to impart the kinds of creative ideas of the engineer into the software? That's a kind of leading either-or question ...

TE: So the answer is yes [laughter]. All of that. There's certainly space for all sorts of tools, and precise tools are really wonderful. They allow you to really control the process and get repeatable results. But even in a precise tool you don't necessarily link the parameters directly to the user interface. There's many internal parameters that you might choose to expose to the user interface; this may be a good thing, but at times it's just overwhelming for a user to see.

Say you're doing something like time stretching, and there are so many things you can do in there: you can change the window shape, you can change the overlap factor; instead of giving it a stretch ratio, you could specify the input and output separately. You could use vertical coherence in the spectra or horizontal coherence and work on those two things separately. And there's lots of different techniques to do that.

So in the design of a tool, you may want to bring up parameters that are more meaningful to the musician – or not. For example, one choice is whether to work with frequency or with note names, when talking about pitch; or in amplitude, you could work in decibels or some other kind of perceptual unit of volume, using some kind of Fletcher–Munson curve, for example.

I think all of those things are valid; they might all make more useful tools. For example, if it's a knob that the user is turning, you want the user to notice things happening, and for that to be consistent with some sort of musicality. For example, in my article I wrote about delay feedback; when you turn the feedback control, do you want to hear the volume decay in each echo, or do you want to hear the number of perceptible echoes decrease? And of course that question is also different when there's a longer delay or shorter delay – when it's a shorter delay, when does it turn into resonance? And then when does the quality of the resonance, its frequency spectrum, become interesting?

So if I make controls that lead the user to adjust the things *I* want to address, that's good for me. Now, for someone who is extremely knowledgeable and sound synthesis, you'd probably want to be more exacting and expose a lot of things. These are all the thoughts that go through my mind when I'm designing something: just what exactly is this knob doing, and how is it going to interact with other elements in the system?

TG: I was curious about your imagined user of *SoundHack* in the early days: were you imagining that it would be used by 'hackers', or perhaps people particularly knowledgeable about computer music? Or musicians and composers who were trained in more of a non-electronic musical tradition, which still for many people had (and has) little to do with electronics and computing? What were the use cases you were thinking about?

TE: This is back in 1991–92, so I think I was thinking mainly about experimental musicians back then – those were the people I sent out notices to. People working in synthesis who were a bit more commercial. I thought maybe they'd be into it, but I didn't think anyone outside of the academic experimenter – or just plain experimenter – would be patient enough to wait for the results. This was non-real-time software on fairly slow computers! They were running 16 megahertz or something around there. It took a lot of patience to use my software.

But I was surprised when I found it showing up in film studios. This was the first time I noticed that there was a potential commercial application.

TG: How did that happen? Did you go to a movie and hear something and think 'Oh my God, that's *SoundHack*'?

TE: No, no, the sound designers would get in touch with me. The filmmakers from the film *The City of Lost Children*, Marc Caro and Jean-Pierre Jeunet, got in touch with me and said 'Hey, we're using *SoundHack* on our soundtrack.' They made these amazing surrealistic films. And then I heard from the sound design company that was working on *The Matrix*, and how they had multiple computers all churning away on the soundtrack, stretching things out, making sounds which were incorporated. I thought, 'Oh, this is nice.'

TG: Was it freeware? Shareware? These are big budget film studios – what kind of licensing were you doing with the software back then?

TE: At first I think I sold it for some amount; and it became 'Musicware' after that, when it became easier to distribute things on the Internet.

TG: What's 'Musicware'?

TE: Basically I told people they could use the software if they sent me music in exchange for it. I soon had more cassette tapes and CDs than I could ever listen to. And after that it became freeware, essentially. I didn't really know what to do with it.

TG: But you still kept working on it, patching it, porting it, supporting it.

TE: I did a ton of creative work on it right after the LA Earthquake in 1994. I was working at CalArts and our whole facility was destroyed! So we had a lot of time on our hands to just get in the studio and work, because most everything else was impossible. I probably kept on working on it until about 2004, and then at that point, I realised I wanted to work on real time things again.

TG: What was it like shifting from software back to hardware? From designing non-real-time signal processing software and plugins into designing Eurorack modules with Make Noise?

TE: At first I didn't really know what to expect; I hadn't really worked on things where you could grab multiple knobs at the same time and patch them live. Certainly I had worked with Max and PD, but it wasn't the same kind of interaction. With Pro Tools and even Ableton Live it's all really lots of preparation in the studio. So this was totally different.

I started out by just designing a delay that did pitch shifting; I thought that was something that was missing in the modular world at the time, so it was an easy one for me to do. I knew I could make it sound good, and there was hardly anything out there like this for people to play with. And at Make Noise, they helped me realise that I had to make this an integrated part of a modular system, the type of module where you're going to want to plug other modules in to it and set up feedback paths, set up control voltage, allow it to be clocked – allow it to really interact with any other module in the system. So that was very different from a plugin effect.

TG: Do the modules you designed for Make Noise use software?

TE: The first module I designed was based on a PIC microcontroller. It wasn't as fast as I wanted it to be, so I programmed the software in assembly to get as much speed as I could out of it. But now we're using different chips – I won't name any manufacturers – but they're very fast. One of the main needs for Eurorack is lots of voltage inputs. If there's a multiplexed analogue to digital converter (ADC) that's a bonus. The other need is, and this is not news to anyone working in hardware, that you want to keep the cost of your components low, because it's just multiplied when it gets to the musician. And you also need to keep the power consumption very low, because people want to plug in lots of modules to their system. These engineering challenges are very different from those in software.

But yes, everything I've designed is almost totally realized through software. There are a few analogue paths; I think the wet/dry mix in the reverb module is analogue. But everything else is digital.

TG: How do you feel this work relates to the 1990s hacker worlds you've also operated within? There's a lot of ideology – some might even say propaganda, if you want to be critical about it – that the Eurorack standard, and contemporary modular synthesis, is 'democratising' in a certain sense – like, 'everyone can build your own system, you can just watch some YouTube videos, learn how to solder, get DIY kits, "hack" your own system together'. I'm curious how you see these two communities or scenes relating to each other.

TE: Well, I'd say the hacker culture of the 1980s and 1990s was basically the only thing around, so it tended not to get into camps. With Eurorack, it emerged out of maker culture, but now it's a commercial thing. And when things are commercial, the musicians that use this or that often get very dedicated to their instrument – I'll often get into discussions where people start right away saying 'well, you know I use *this* because of *this*', and they have to justify themselves, and there's a bit of competition between users of different brands.

Hacker culture moved to maker culture when it became easier to make your own hardware. I mean you could always make your own hardware, but at a certain point it became possible to easily make and replicate and make your own panels or circuit boards – you didn't have to look hard to find a specialist, there are online companies that do this. That's what made the whole Eurorack thing possible, all of these hobbyists becoming builders and then becoming commercial.

TG: And all this was reliant on the global electronics infrastructure – the availability of streamlined PCB layout and manufacturing making things incredibly cheap for people in America and Europe.

TE: Embedded processors are really wonderful – I mean we did DSP stuff in the 1980s and 1990s, but we were using these Motorola DSP56000 chips – expensive chips which were great at the time, but they were also considered possibly dangerous – I remember having to sign a waiver to get them at one point.

TG: How were they dangerous?

TE: Well, because you could use them in weapons.

TG: This brings me back to your comment about driving around the East Bay looking for surplus technology – one reason it was there was because of this massive top-down investment in defence and defence-related research in aeronautics and computing, so of course there's going to be technology there that becomes 'surplus' when it isn't useful anymore for the people who funded its development and who were the main purchasers, say, 10 years prior.

TE: Yes, definitely.

TG: Another historical connection between the 2000s and the 1990s that I think is interesting is the fact that you started out writing non-real-time software which turned into *SoundHack*, and in your article you draw a line between that and the work you've been doing with Make Noise. To a casual observer, perhaps, there might not be an easy or obvious connection between these things – non-real-time software, software plugins and Eurorack modules. How do you draw this line from software to hardware – and then essentially back to the software that's running in your modules?

TE: There's a lot of ways to answer that. As far as the culture of Eurorack, the aesthetics tend to be very open and experimental. And this is the space where I like to be in – these are the kinds of sounds I like to make, and it's very fun interacting with this community by designing things and surprising people – making boxes that can go from overly subtle to overly noisy in one knob movement. In both worlds I'm not tied down to commercial expectation in my designs or to commercial music expectations for the kinds of sounds they make. Of course the experimentalism of today is the commercial music of 10 years from now, but right now that's where I like to be, and it's the new sounds that interest me.

The main advantage of modular, from a user point of view – and here's where I'll probably get the most argument – that from the user point of view, you have an instrument where you can freely change all the parameters physically. You can freely perform with it, and that is really interesting to me. Plus, you have voltage control, and you can program it so it can satisfy various ways of working with musical devices. There is the notion that it's separate from computer practice, but I don't know if I agree with that. You can hybridise it, you can connect your computer to it – there's lots of ways of working with it, so I think it's quite free.

As far as being the developer of these modular devices, what I like most about programming hardware is that I am in control of the computer. Writing plugins and interacting with companies like Avid and Ableton is actually not bad at all; they're usually very generous and helpful. But what's more difficult is the rate of revision of operating systems, and how things you rely on disappear every year, and that can be a little tiresome. And if you really want to make plugins seriously, you want to get them out to lots of different people – using Windows, using macOS, different kinds of plugin formats – so you end up spending far too much time maintaining your software without working on new ideas.

TG: This seems to be a really big irony – that in an era where computing power becomes cheaper and cheaper, and devices become more ubiquitous, supporting software ends up being impossible because companies need something new to sell every year.

TE: A lot of companies are under pressure to make one major version revision every year, to get the users excited and subscribed to the new version. I mean, I'm not totally free from hamster wheels, and you do get caught up in the idea that you can always revise and make your old work better. But if you can keep that creative impulse, and even join it to that kind of work, then it works.

TG: In the early 1990s when you were developing *SoundHack*, what was the financial outlook like? Was it a commercial product?

TE: Well, I didn't make that much money. For a while Frog Peak Music distributed it on floppy disk. I basically thought, 'if we're going to distribute *SoundHack* on floppy disks, on physical media, maybe we should sell them. But I had a job at Mills College, and then after that at CalArts, so I didn't think I needed to get paid for *SoundHack*.

TG: I wonder if there are still copies of those disks in somebody's garage ...

TE: Oh yeah! I see them show up on Instagram every so often.

TG: How do you view the current landscape we're in, where there is such a diverse proliferation of ways that people are exploring the kinds of processes you were interested in *SoundHack*? In the studio in 1989, you were frustrated at the lack of standardisation for digital audio files, the inability to run early environments like *Csound* and execute processes like convolution and phase vocoding with the computers you had at hand. And now, all of that is possible in both software and dedicated hardware, essentially in non-real-time and in real-time.

TE: One of the hard things these days is creating your own way of working. There are so many tools out there and so many interesting things happening in each one of these tools. People who are working in Supercollider are getting this whole different realm of sound as compared to people working in PD, or people working in Ableton. They just lead to different spaces. But also, the developers in all of those formats have interesting things going on. So, dedicating yourself to one sort of way of making music or format actually seems totally fine to me.

Of course there's always the sense of looking over your shoulder and thinking, 'Oh, this person just got a boutique handmade computer and I want to do that too...' we're at this point where there are so many ways of creating synthesis and making music, and they're all very tempting.

TG: There's a certain amount of path dependency that develops, and silos that form around communities and practices.

TE: Well, that's not bad – as long as we keep going to each other's concerts!