Larry Polansky on HMSL and Computer Music.

- Interviewed by Alistair Riddell
(Melbourne 2/8/89)

AR Larry, you are primarily known as a computer music composer because of your involvement with HMSL, what is your background in both computer and electronic music? When did you first encounter the computer?

LP When I was 18 and in my first year at college. Until then I was primarily a jazz player, also playing traditional American music. I was a gigging professional musician but eventually decided to focus my musical energies on composition, although I still perform a great deal. I first encountered the computer as a mathematics student. Mathematics is important to me in my understanding of music, and I actually got a degree in math, concentrating mostly in pure math. I see that now I was interested in form, and that early interest in topology and set theory seems to me to be very akin to my current interests in musical form. My first programs were simple little "AI" experiments: counterpoint writing programs and the like on a PDP 11/45. This was at a small college in Florida.

AR Did you work in that area when you went to Illinois?

LP No, I went to Illinois several years later. After I left Florida, I met James Tenney at the University of California and started working with him.

AR Tenney, as I understand, had been quite involved with computers.

LP Tenney is a very important figure, although not as well known as I think he should be. He was, for example, instrumental in adding algorithmic power to the early digital synthesis programs at Bell Labs. I believe that Jim (and others of course) encouraged Max Matthews to add provisions for writing one's own compositional subroutines to generate score files. Tenney, although also interested in timbral experimentation, said, more or less: "well, what we really want to do is compose." I think his interest in using the computer as a compositional mind was very much influenced by his association with Cage. He saw the power of the computer as a compositional and perceptual modelling tool. In the early 70's Tenney and I worked on a computer program to model hierarchical temporal gestalt perception. The results of this project were later published in the Journal of Music Theory, and a few other places. I still think it was and is a quite extraordinary program, even though it's quite small (and of all things, written in Fortran). But we worked for about two years in refining it, to make the program a very integral model of the theory, which was primarily Jim's. It was a great experience for me, to work with someone like that so closely, for so long.

Around that time I also worked in many different types of technology. I worked at Stanford, and I worked a lot with computer controlled analogue equipment. Basically, whatever I could get my hands on I used. At the University of California at Santa Cruz I worked with Bob Hoover, who later became founder of 'Mimetics' and responsible for the Amiga Soundscape software. He and I interfaced an old U.S. navy reject Interdata Model 3 computer, and had it controlling Moogs, and even did some live performances with it (I remember Gordon Mumma somehow magically coming up with a truck for us to move this thing in!). This kind of work was all machine language programming, with no mass storage, and very slow. In those days, the early 70's, there wasn't much around in terms of computing power (at least for me), and one had to be fairly resourceful. You could work very hard and produce very little, but you learned a lot and it was certainly fun. Of course, I've worked on a lot of different systems since then!

AR You have also been paralleling your work in electronics with writing for traditional instruments.

LP I really like writing for instruments and I like working with performers. I have never been exclusively interested in electronic sounds, and this is perhaps some sort of artifact of my interest in compositional artificial intelligence. I do work with various synthesis techniques, and have done a bit of experimental work in that area, but that has never been my area of speciality. I'm much more interested in generative, formal, and compositional processes (although sometimes, happily, these two areas of computer music intersect). Much of my computer music work results in instrumental works. Strangely enough, this focus may be shifting. I've recently been working on a synthesis algorithm which stems from what was essentially a formal process. But mainly I've worked with whatever sound producing means have been immediately available, and often they have happened to be performers. Many of my close friends are great performers, so I like to work with them. For example, I wrote a computer generated flute piece

"I do work with various synthesis techniques, and have done a bit of experimental work in that area, but that has never been my area of speciality. I'm much more interested in generative, formal, and compositional processes..."

(*V'leem'shol*) for Ann Laberge, and there's the tap dancer pieces, things like that. With someone like Ann, you hear her play, and you have to ask yourself "How can I NOT write for someone like that"? It really goes both ways. Most of my recent music has been more or less for solo interactive computer and one performer (like *B'rey'sheet*, which I do with my wife Jody Diamond, and which we did at the Astra Choir concert in Melbourne in August. On that concert I also did a duet with Chris Mann, which will be released soon on an Artifact Records CD). I like this situation a lot. It seems to integrate many of my interests. I get to work with my performer friends and also do live computer music. The synthesis aspect has been a hard one for me to deal with. On the one hand I'm not really that attracted to tape pieces and

From Chroma: Newsletter of the Australian Computer Music Assn.
Number 3+4, December 1989

that seems to me the main area of really interesting synthesis up to now, especially if you are not, as I am not, a hardware genius. If one were really committed to the most powerful synthesis techniques, it seems to me it would be difficult to reconcile that with a commitment to live electronic performance at this time (although with all the new DSP developments, this is certainly changing!). Unhappily for me, and I think many other composers find themselves in this situation, I've had to accept a lot of sound producing stuff which I didn't build or design myself. I've always concentrated on software design and theory, but it's not a comfortable situation for me to use a lot of other musical stuff designed by other people, especially those with more commercial intentions. I've been pretty successful at doing some odd things with pre-existing gear, but I don't really think that ultimately this is satisfactory. So great performers for me have been a kind of a way out. They are a way of using a sound producing means that is accepted, and I think pretty honest.

"I really feel that I am part of a community and I think we're all in danger of being kind of precious about our work. I think that's an old way of thinking about music. One way to help usher in the millenium perhaps is to acknowledge that community, and it's beauty."

I want perhaps to take this notion of sound one step further, because I think we're at a point where we're all going to change our thinking. For me, I feel like I can personally begin working seriously in the combination of live computer work, software development, and synthesis again, because of all the fantastic developments in small cheap signal processors. The 56000, TMS320-x0 series, and other chips are making it possible to do very interesting things in in more or less real-time. In fact, several composers, like Tim Perkis for instance, have been working in this area for some years. And of course they're getting faster and faster. The NeXT machine is also changing a lot of minds. Admittedly it's still expensive, but not like a VAX. Even the Amiga was a real revolution to me. Four channels of DMA sound. Ok, it's 8 bit, and sounds pretty funky but you didn't have to solder and you didn't have to design circuits, and you got a very hands-on access to the waveforms themselves. I've been able to do a lot of interesting experimentation with that aspect of the machine, and people like Robert Marsanyi have done some extraordinary work in this area.

AR In looking over some of your compositions I noticed that they're all dedicated to people and that you acknowledge them and their ideas a great deal in your work.

LP Yes, although I'm very positive about my own work, I'm not shy about references. In fact I believe very strongly in our interconnections. I really feel that I am part of a community and I think we're all in danger of being kind of precious about our work. I think that's an old way of thinking about music. One way to help usher in the millenium perhaps is to acknowledge that community, and it's beauty. When I do pieces like the *Distance musics* (published in 'Perspectives') where every piece is a tribute to some other composer (but I suppose, ultimately, very much my own) I am trying to acknowledge that very heterarchical intellectual and musical community. I think a lot of composers are shy about acknowledging influences, they say "well, that's MY own idea" ... and so on, like musical ideas are some kind of possession. But I feel part of a community of mind – I WANT to be part of it and I WANT to help engender it. Again this probably comes back to something like HMSL, it's very much a group action.

I think that if one really looks at say, the early work at the San Francisco tape centre, composers like Don Buchla, David Rosenboom, Tony Gnazzo, Ramone Sender, and especially the League of Automatic Music Composers, were all working towards a musical community – and that's not a dead idea, it just somehow got overlooked in certain areas of our musical environment. The technology wasn't quite there in the sixties for certain experiments to this end (although I think one could say that David Rosenboom's biofeedback work is pioneering in this respect) but now it is! Now we're on BITnet, we're sending each other discs, we're doing lots of interactive and communicative pieces, we're working on code collaboratively and, of course, these methods and ideas will evolve and change radically and wonderfully in the next five years. We can't even imagine what those ways will be. Perhaps we'll be sticking electrodes on our heads and thinking pieces. I like that a lot. I'm of course not originating these ideas, but I am part of a lot of this work, and I'm grateful for this.

AR Could you tell us something of the history of HMSL?

LP David Rosenboom, James Tenney and I had been very close friends for many years. We had worked together in various ways and shared similar ideas about form, transformations of forms, recognition of forms and computer modelling. We shared an office together in 1976 while we were writing that "perceptron" program I mentioned earlier. David was writing some similar programs of his own as part of his 'On Being Invisible' series, to analyse responses from the brain.

So it really has a theoretical underpinning from three people who were thinking about some common issues. I think Jim is the theoretical godfather of the whole thing. His work in this area goes way back to 'Meta-Hodos' in the early sixties, and his insights still exist in various forms in HMSL. I think if one looks at David's earliest electronic music, one also sees a deep concern with the idea of languages. This evolved quite naturally, I think, into a concern for language environments for composition and performance. He participated in the development of the Buchla/Crowe language, Patch-IV, which was a terrific hybrid control environment, and also wrote the language FOIL (Far Out Instrument Language) for the Touche, which he built with Don Buchla. We began thinking about implementing a very general and very powerful language for small computers that a lot of people could use in radically

diverse experimental situations. Our interest in this, combined with our common theoretical bent and the advent of 16-bit microprocessors in the late 70's, led to the inception of HMSL. I think David saw HMSL as the next generation after things like 'PatchIV' and 'FOIL', and it really does come somewhat from that tradition of flexible experimental environments. Of course, it's become a LOT more than that, but in my very earliest prototypes I was interested in abstracting, for example, things like the notion of definable stimulus/response events that was so wonderful in PatchIV. David brought me to Mills College (Center for Contemporary Music) in 1980/81 and we set to work on writing HMSL. We built a 68000 S-100 system for the prototype. We also started the Seminars On Formal Methods, which were focused on formal systems of thought about music and language, and we tried to get lots of interesting people to Mills so that we could all talk about these ideas and work with together. Dan Kelley, for example, was just starting to think about 'Masc' at the time, and in fact 'Masc' and HMSL share some basic routines. We were trading Forth routines with lots of people at the time. One spring we brought Ron Kuivila to the CCM, while he was working on the first version of Formula, and I was playing with some simple HMSL scheduling ideas. Ron contributed some very powerful and interesting code and ideas, and so did of course, many others. We were all sharing ideas. There seemed to be something fascinating in the air at that time, people thinking about experimental music languages because the technology allowed it. Something we couldn't do in quite the same way before.

One of the reasons we picked Forth was that it was kind of a *lingua franca* for small computer music users, especially in the Bay Area. Many composers who did live stuff, like David Behrman, George Lewis, Joel Ryan, John Bischoff, and others, knew it, used it and liked it. It looked like it was going to continue to be very important in that area. David envisioned the CCM as a kind of 'language clearing house' for this kind of work, and it actually was to a great extent. It was someplace people could come to and trade ideas, and try out new things. It was VERY active and VERY busy in those days, and also a lot of fun. Of course, it's STILL busy and active! We wrote the prototype of HMSL in those first couple of years. Much of the time went into the design, simply thinking about what things like the data structures should be called. I wrote the prototype on the S-100 system in Forth with a lot of help from friends like Dan Kelley and Phil Burk, who were just sort of hanging out at the Center, hacking away, and helping me through some gnarly operating system problems.

Phil Burk started hanging around the Center because we were the only people in town with a 68000 running that he could experiment on. This was slightly before the days of the Macintosh and Amiga. He just wanted to play with it. Phil is a computer genius who used to build Z-80 systems, as I used to say, from the body parts of small furry animals in his basement. He was really thrilled about playing around with the 68000 and he was an invaluable aid. We became good friends, and he's such a fantastic programmer and brilliant thinker that he was a natural to add to the HMSL 'team'.

About two years later when the system was more or less up and running at the Center but a bit clunky and hard to use (for example, since it was a 16-bit Forth, you could only have 64k of code!), we were fortunate to be able to bring Phil into the project as a full third design partner. His first contribution was to say "Look, there is this thing called Object-Oriented programming out there that we can probably use" We had actually seen 'SmallTalk' but Phil had been working professionally on a very early compiler for the Mac called 'Neon' and he was very enthused about Object-Oriented programming. He also recognized the natural affinity between the ways we had designed the language and the concepts of OOP languages.

"Phil [Burk] actually wrote the first version of HMSL in Object-Oriented code on a Commodore 64 (!) because that's what he had at home."

Phil actually wrote the first version of HMSL in Object-Oriented code on a Commodore 64 (!) because that's what he had at home. MIDI didn't exist at the time, but when it came we MIDIfied the system very quickly; that wasn't very hard. Phil wrote an Object-Oriented version for the system at the Center, and that system was used for Pauline Oliveros' *Dear John* (a work for John Cage's 75th birthday, commissioned by the West German Radio) many of my own pieces, some graduate student works, and others. But since this was an S-100 system, it wasn't really portable and was running some really old fashioned technology, like a stand alone S-100 graphics card. Even though there was MIDI, it was tied into a Buchla 400 digital oscillator system - for which we had to kind of hack the interface. Phil used to joke that it was pretty portable – anyone who had an ERG S-100 based 68k system with a Buchla 400 oscillator card could run it! Then the Amiga came out. The Mac had been out for a short time but it was still new. The original Mac's were difficult to program on, and it was clear that there would be some rapid developments that would make them more accessible. We got very excited by the Amiga though – it was cheap, had a nice operating system, lots of power, was fast, and all kinds of interesting features.

Phil became immediately involved writing a Forth compiler called JForth for it; he was a big Forth fan at that time. So he did that, and we got developer status on the Amiga, and within a month or two he had a version of HMSL running on the Amiga, and all of a sudden there wasn't 64K of memory to play with but a couple of Megabytes. Amazingly, I was able to more or less transfer the piece *B'rey'sheet* from the ERG to the Amiga with very little revision.

For the first time HMSL became a good environment that a lot of people could use. That was Version 2.0, which we distributed to various people (like Nick Didkovsky in NY) for Beta testing. The response was very enthusiastic, I think, because it was so 'hardcore', and a lot of composers

(like Nick for example) were looking for flexible, powerful environments, and weren't really too put off by the difficulty. In fact, they enjoyed it! We wanted to get a lot of feedback – it was buggy but sort of worked. I also started doing pieces in it, and Phil Burk, Phil Stone and I actually did what may be the first concert ever done entirely with Amiga local sound, in San Francisco. We used HMSL, and had the three computers communicating in various ways. It was strange and very interesting to me! At the same time Mills College became committed to Macintosh technology, and we decided that our strategy for HMSL would be to support a Mac/Amiga parallel. We ported HMSL to the Mac, which took some time because there was a different Forth compiler for the Mac and back then, the Mac operating system wasn't easy to figure out. The port was tricky and consumed Phil for quite a long time. But it worked and HMSL has been more or less free and clear since then.

From that time on we've been concentrating on developing it further, distributing it, teaching it, using it, writing about it, and documenting it. It's been quite a project! It's still changing radically. Version 4.0, which will be out soon, is very different and far superior, I think, to the previous versions. It will have a whole new graphics system, lots of sophisticated MIDI support (like some interesting sequencing stuff, MIDIfiles, user-definable patch editing, a score entry system), and some nice refinements to the data structures themselves which will make them more powerful and I think easier to understand. We're excited about it of course.

"...we got developer status on the Amiga, and within a month or two he had a version of HMSL running on the Amiga, and all of a sudden there wasn't 64K of memory to play with but a couple of Megabytes."

AR On reading the various articles about HMSL that have appeared in recent years, I was struck by both the terminology and conceptual model that has emerged from the work at Mills college. It seems to me that new users will have to confront this before they can mould HMSL to their own way of composing. In other words before they can reject any part of this work they would have to know it fairly well. Is this more or less the case?

LP Well, they certainly would under some circumstances, but they do not necessarily have to get involved too deeply with that part of HMSL. It would be possible to just use HMSL as a programmable MIDI or video generator, although I think there are other systems that can do that as well. But we were interested in the fertility of the system. We wanted to create a deep environment where people could take our technical and philosophical ideas further, or reject aspects of them. HMSL is distributed as source code and it is very well documented. You can carve up the system as deeply as you like, and we'll help! We'll tell anybody anything. If you wanted to rewrite the scheduler itself we'll tell you how to do it. We're very open about it. The idea is that people will do that – make it their own. Many have done exactly this sort of thing, and that gives us a lot of gratification.

AR What is the future of HMSL at the moment?

LP Well 4.0 has to come out. We'll keep distributing it and supporting the community aspect of it. I don't think there is a danger of it becoming obsolete within a few years. I also want to become the most active user of it! After all, I designed it to make the kind of music that interested me, and now I figure I've got a right to take some time and actually make some of those pieces.

Further Reading:

Polansky, Larry, and David Rosenboom. 1985.
"HMSL (Hierarchical Music Specification Language) A Real-Time Environment for Formal, Perceptual and Compositional Experimentation." In *Proceedings of the 1985 Computer Music Conference*, ed. Barry Truax, pp. 243-50. San Francisco, CA: Computer Music Association.

Polansky, Larry, David Rosenboom, and Phil Burk. 1987.
"HMSL: Overview (version 3.1) and Notes on Intelligent Instrument Design." In *Proceedings of the 1987 International Computer Music Conference*, ed. James Beauchamp, pp.220-27. San Francisco, CA: Computer Music Association.

Polansky, Larry, and John Levin. 1987.
"The Mills College Center for Contemporary Music's Seminar in Formal Methods series: A Documentary Survey." *LEONARDO*. (Special issue on Visual Art, Sound, Music and Technology), ed. Larry Polansky. Vol 20, No 2. pp.155-64. Pergamon Press.

Polansky, Larry. 1987.
"Paratactical Tuning: An Agenda for the Use of Computers in Experimental Tuning." *Computer Music Journal*, 11:1(61-68).

Tenny, James. 1961.
Meta-Hodos and META Meta-Hodos, 2nd Edition 1988. ed. Larry Polansky. Oakland, CA: Frog Peak Music.