Code and Music: Technology and Creativity in Composing

Jonathan Norton

Since the first singing of "Daisy" echoed through the halls of Bell Laboratories from their mainframe, computers and their software code have become an integral part of how music is created and, for better or worse, distributed. In today's world there is almost no area of music that has not been touched by technology. As Gerfried Stocker has observed, "Artistic expression no longer arises solely as the result of artists' talent and mastery of their craft, rather, program codes describe parameters and processes in accordance with dynamic visual, and sound environments are generated via computer … serious and popular alike."

(Ars Electronica Festival, Press Release, Linz, March 25, 2003.)

The symposium has asked me to consider the role of code in music. Since this is a daunting task, I will examine briefly how code touches and even shapes certain aspects of music composition, and pose some questions for exploration. We cannot put the genie back into the bottle, nor do we wish to, but we should think about the way code has been fused into art and music to free our own creativity.

A specific question was posed recently in an Ars Electronica press release this Spring: "How do software and digital codes impact the essence and identity of media art as 'art created out of code', that is, as a generative and processual art form that has developed from and consists of algorithmic and computational processes?" This question is very much on target.

Some of the larger changes have occurred in algorithmic composition. Although lower level programming environments based on Lisp, C and C++ still exist and are still commonly used, such as Common Lisp Music (CLM), Csound, and the Synthesis ToolKit in C++ (STK), they are mainly the playground for academic institutions that cater to research and the further development of the genre. Only within the last several years has this begun to change. The computing power that was once only available to these institutions a mere ten years ago has dramatically changed. Today the typical home computer is powerful enough to run just about any programming environment with room to spare. This has unlocked a whole new market of potential users.

Along with less expensive and faster computers, the advent of higher level GUI programming environments such as PD and in particular MAX/MSP, have allowed more people access to computer music than ever before. Since it is no longer only the realm of academics, the mainstream now has a chance to enter the domain of computer music. Opening the doors to the more mainstream also yields the side benefit of innovations developing more quickly. Composers, academic and non-academic alike, who once hesitated to attempt to create computer music, due in part to lower level languages, are now able to let their imaginations soar in an entirely new area of music. This has profoundly changed the musical landscape.

For example, about nine or ten years ago when I wanted to write algorithmic composition I mainly used Rick Taube's Stella program which was based on Lisp. It was a powerful program tailor-made for this type of composition that was relatively easy to learn for anyone with knowledge of Lisp. MAX was available at that time, but its capabilities were limited as a more general programming environment. Both Stella and MAX, however, were limited because they could only use MIDI. When I wanted to write music that entailed computer generated and manipulated audio I would have to switch to a program such as CLM. Yet, Stella and CLM environments still required writing many lines of computer code that ran on the NeXT computer. Today, if I want to create this type of music, I normally use MAX/MSP, which now combines the qualities of algorithmic composition and audio together in one package, and it runs on my laptop.

With the help of faster computers and GUI programming environments, another area of rapid growth is in the field of musical controllers and interfaces. In fact, it has become so popular in the past few years that a new international conference has sprung up to address the topic—the New Interfaces for Musical Expression conference (NIME) held at McGill University in Montreal, Canada this year.

Many of these controllers are cross-pollinations from different fields that take objects from their normally intended environment and use them in a novel way. BioControl's BioMuse was one of the first systems to employ bioelectric transdermal electrodes—normally used for biofeedback and medical therapy—for musical applications. The system could measure electromyograms (muscle movement), electrooculograms (eye movement), electrocardiograms (heart), and electroencephalograms (brain waves) from eight different sensors at once. While this was great at the time, its application was pretty much limited to academic use and composition because of lack of knowledge among and, thus, access to those outside of academic and medical circles.

More recently, Infusion Systems' I-Cube and similar products have entered the scene. The I-Cube system does not work with bioelectric sensors like the BioMuse, but has a more flexible design that allows the use of up to 32 sensors at once. It also features a wider array of sensors that measure such parameters as bend, touch, light, tilt, magnetic field, orientation, temperature and pressure. An important advantage and time saver of these systems is that they automatically translate the raw sensor information into MIDI data. This in and of itself makes them much more user friendly. When combined with a GUI programming environment they lend themselves well to reaching a larger population due to this environment's accessibility and ease of use.

Not only do the controllers and software code allow for more organic and evolving sound, but they also allow for the sound to evolve in a real-time performance setting. This new possibility was not available in the past or at least not as easily accessible. Previously, in a live situation, sound had to be physically produced either by miking an instrument or played from tape before it could be altered through EQ, filters, tape manipulation, or the use of many outboard effect boxes. However, the sound source itself could not adapt. Currently, the sounds themselves can be completely adaptive and the parameters that affect them can be spontaneously produced on the command of the performer.

So, in the larger context of music with all of these advances in technology, how has the position of the artists changed by their use of software? For traditional composers it means that they no longer have to write music with pencil and paper only. While traditional pencil and paper are still used, software has allowed them to create, audition and generate music and musical scores faster, with more diversity, and in more directions than ever before. Sequencer programs make it easier to audition different textures and arrangements before committing oneself to them. Most of these programs also allow the integration of audio files into the sequences, which further enhances the composer's palette of possibilities as never before. Notation programs, once learned, are also invaluable time savers. Entire sections of music can be entered, altered, cut and pasted, transposed and checked for range, all with relative ease. Once completed, the individual parts can be extracted and

printed automatically from the score. For orchestral scores and other large works, notation programs are worth their weight in gold.
For modern composers, the proliferation of software choices means that they have become part composer and part programmer. In addition to digital audio, sequencer programs can import a plethora of plug-ins for effects, sound manipulation, samplers and virtual instruments, just to name a few. For composers who are not satisfied with the limitations of existing programs and plug-ins, high and low level programming environments are the tools of choice. This is the realm of algorithmic composition, acoustic models, physical models, digital signal processing and unique controllers. Here, every parameter and nuance of sound can be dictated and tailor made for specific situations, sonorities and musical ideas. Of course this flexibility comes with a price. The more nuances composers want to control, the deeper their understanding must be in terms of synthesis techniques, acoustic modeling, digital signal processing, and how the parameters of each affect one another.
For artists in other disciplines, software has allowed them to become composers and vice versa. In the past, without software a person truly had to be a composer. In order to write music, pencil and paper were needed to create a score and then shown to a musician to be played. This required certain mastery at the very least of how to notate music as well as knowledge of the capabilities and limitations of the instruments for which the pieces were being written. Through the use of software, hardware and computer-generated or computer-controlled sounds this is no longer the case. Most art in one form or another can be translated into almost any other art form. An artist can take an image of his/her art, be it a photograph, a drawing, or a form of video and, through a scanner or video camera connected to translation software, map the pixel density or rate of movement into raw MIDI data. This data in turn could be sent to a synthesizer or any other sound source or programming environment to create sound. Another example would be a sculptor who uses light, heat, touch or bend sensors attached to his/her sculpture to generate raw MIDI data which is patched to a sound source. The possible combinations are endless.
Conversely, a composer could take an existing composition or a live performance and through the use of MIDI to video or audio to video converters create a graphic representation of his/her music in real-time based on some predetermined parameter mappings.
As I conclude, I come back to specific concerns with code and creativity. As stated above, most modern composers and especially computer music composers have become part composer and part programmer. Some people might say that such a composer's music is not musical, but most people will not argue about the music being a form of art. But what about the code used to produce the music? With an increased reliance on computers and software is it possible to say that the code itself has become an art form in its own right? If so, at what point does software stop being simply code and cross the line into art? Do these rules change for different creators or do they stay the same across the board? For example, if a computer programmer writes, say, an algorithm that generates music, is that program considered to be art? What if on the other hand a composer writes the exact same algorithm? Is the composer's program considered art and the computer programmer's program seen merely as code, as a means to the end product of music, or are they seen on equal footing? Is either one considered art? Is a program that has been labeled as art truly art or is it just merely because someone says, "This is art?" Are there any concrete criteria on which to base such a decision on or is it merely subjective? These are questions that move us toward the heart of the integration of technology and art.

Although I have only talked about code, technology and music composition, the concept of code and music has many levels that will probably affect us all in the future. As I mentioned in the beginning, computer technology and software have become an integral part not only of how music is created, but also how it is distributed. The current controversy about the distribution of music and file swapping through the Internet involves law, code and individual creation of art at an even broader level. Technology and code that broaden the art of individual creativity, as I have discussed, also threaten to change individuals' rights to their art.