

Shane Hoversten

Mary Farbood: Hyperscore: A New Approach to Interactive Computer-Generated Music

This paper was about the development of Hyperscore across four versions. Hyperscore is a music composition program of sorts; I say “of sorts” because the exact nature of the composition was not static across the versions, and in some of these versions resembled a music generator more than a composition tool. Farbood, the designer of the program, seemed to wrestle with the question of how much, exactly, it should do for the user. This is a question that transcends Hyperscore and the music composition domain. One could ask it with regard to any tool designed to further any creative pursuit.

Hyperscore’s fundamental abstraction is mapping a human-drawn line onto some components of a musical score. In the first version of Hyperscore the human provided only one line, and selected some combination of nine musical chunks, or motives; the program interpreted characteristics of the line and “annotated” it with four other tracks, whose exact characteristics were based on the motives the user selected, or an algorithmically-determined number of rests.

The second version of Hyperscore maintained the abstraction of a principle line, or tenor, but this time the user was allowed to annotate the line himself, again using the drawing abstraction, by making a series of “exclamations” near the main line using different colored “pens.” The pens correspond to the musical motives that Hyperscore itself utilized in the first version. The graphical characteristics of these user-created annotations influenced the accompaniment that Hyperscore provided. Also new in this version was a harmonic generator, which used a hierarchical HMM to come up with harmonies. The parameters of this HHMM were hand-tuned by Farbood to produce aesthetically pleasing results.

Farbood mentions a difficulty with the second version in particular, but also with the first version: it was difficult for the user to grasp exactly what Hyperscore was doing to generate music based on the line that was drawn. The third version of the program addressed this criticism. “Instead of *influencing* the decision-making process of the computer, the annotations deterministically *dictated* it.” Now the annotations determined, roughly speaking, the beginning and end of the accompaniment. Hyperscore would loop the selected motive from the beginning until around the end of the annotation. The precise locations of these annotations was also made more explicit to the user via little black boxes, attached to the main tenor line. This allowed the user a finer granularity of control in manipulating these accompaniments. Moreover, the nature of the accompaniment was made to depend more rigidly on the annotation. For example, the length from the main line determined its pitch. This version also added several more harmony types to choose from.

The fourth version of Hyperscore introduced more changes; the user’s control over the strokes was made more precise: instead of forcing the user to alter curves by re-drawing them, she could deform them by dragging. This allowed easier iterative development of the score. Also, the harmonizer was made more intelligent, by incorporating the SPEAC

system for chord progression. SPEAC encodes structural characteristics of harmonic motion. Also, motive transformations have been mapped onto particulars of the graphic annotation; the details of this weren't provided in the paper, but one can imagine the user making a line with a trailing "flair" mapping to motive diminution, for example.

Hyperscore was also discussed as part of the "Toy Symphony" which utilized a number of new musical abstractions in service of composition and production. These included beatbugs, music shapers, voice transformers, and a modified instrument called a Hyperviolin.

All of this work amounts to a thrust in the same direction: by transforming musical production and composition from the manner in which it is traditionally accomplished into a simpler and (possibly) more intuitive one, the process can be made more accessible and more enjoyable, at least to the majority of people who don't have, and don't want to acquire, a deeper, more sophisticated "nuts and bolts" understanding of how music is produced, composed, and represented. This strikes me as the same issue one encounters in software development: one often does not want to concern oneself with what's going on inside the machine. Writing in Lisp, one can deal with very abstract features of the problem space; considering the nature of the problems one habitually faces, this is almost always enough, since subverting one's thinking process to assembly language is a slow, tedious, and error-prone affair.

I don't have an opinion on what is the "natural" level of abstraction for music production and composition. I suspect that the traditional way is not the equivalent of assembly language in my previous analogy. Is it Lisp? Maybe so. But a lot of people find even Lisp too intimately tied to the algorithmic precision the computer demands. If I could, I would happily offload some of this sort of management to a software composition tool that could, with a bit of guidance on my part, do the right thing.

This is where the analogies really part ways, I think: in music, a broad number of Hyperscore's solutions are probable suitable. In software development we are usually considerably more constrained. Still, any tool that enabled you to "zoom out" and deal with the construction process at a higher level would be a useful one, whether the process in question was music composition, software engineering, or hanging drywall.