Applying ImprovisationBuilder to Interactive Composition with MIDI Piano

William Walker
Research Scientist
Advanced Technology Group
Apple Computer, Inc.
walker@apple.com

Brian Belet
Assistant Professor
School of Music
San Jose State University
bbelet@sjcvm1.sjsu.edu

Abstract
In this presentation we will show how the ImprovisationBuilder framework was adapted to our compositions for pianists and computers. Software design issues include: capturing, transforming, and realizing MIDI piano data, and an easy-to-use graphical interface for altering parameters during performance. Compositional issues include: the causal compositional aspect of the ImprovisationBuilder output, and varying the small-scale event and gestural levels while preserving the large-scale structure among multiple performance. The presentation will discuss the compositional and performance importance of these issues and show how each of these issues was handled within the ImprovisationBuilder framework.

Work on this project suggests the advantages of addressing compositional and software design issues simultaneously and cooperatively. Both composer and software designer can benefit from this collaboration.

1. Introduction
We first collaborated on ImprovisationBuilder (IB) during a computer music workshop in 1993, a collaboration we resumed in 1994. Our initial efforts were concentrated into an intense, full-time two week period. Belet had composed some material for a pianist and wanted to design an IB configuration that transform the material in interesting and appropriate ways. This material formed the basis of Belet's (Dismembered) Reflection (Belet, 1994).

We worked together to create some new software components, such as one for creating elixis from sustained single notes and another for adding short ornamentation notes to the attack of single notes. As we wrote and debugged these algorithms, we also gained a greater understanding of IB's pitch representation, while Walker came to understand how Belet intended these musical transformations to relate to the music he had composed for the pianist.

This foundation helped us to communicate when we resumed work in 1994, culminating in our premiere of Cross-Town Traffic, a piece for two improvising pianists and IB in November of 1995. Each pianist's improvisation is transformed by a copy of IB. The transformed material is realized on the other pianist's piano while he is playing. While each of us used a distinct IB configuration, our mutual understanding of the implementation and musical consequences of each other's IB configuration greatly aided our improvisation.

2. ImprovisationBuilder
ImprovisationBuilder(Walker, 1994; Walker, Hebel, Maritirano, & Scalzi, 1992) is a framework for building computer improvisors. It is written in ParcPlace ObjectWorks Smalltalk-80 on the Apple Macintosh. Smalltalk-80 primitives written in C compiled IB to the Yamaha Disklavier and other MIDI devices. MIDI connections are handled by the Apple MID Manager, which sends data through the Macintosh serial port to a MIDI interface.

The IB framework provides Smalltalk-80 classes corresponding to the tasks performed by improvising systems. Listeners process the incoming music, parsing it into phrases and focusing the system's attention. Transformers and Composers create new phrases, either by transforming phrases captured by the listener or by some compositional algorithm. Realizers express musical ideas appropriately, both through real-time presentation and by controlling Timbres that represent sound generating hardware. Improvisory engines the information gathered by Listeners and use it to track progress through a map of the shared musical structure.

Walker & Belet 386 ICMC Proceedings 1996
By providing a 32-bit hierarchy of components for
improvising systems, JB facilitates rapid testing and
prototyping through the combination of existing
components [Johns & Fote, 1988]. Users can
extend the framework by adding their own
custom components to it. These new components reuse
code provided by their superclasses and only implement
the unique aspects of their behavior. JB also provides a
standard representation of musical events for use by
all components.

ImprovvisorComponents are combined in a linear
chain, with a Listener at one end and a Timbre at the
other. This chain of components is contained within
an Improvisor, along with all the information
the computer uses to improvise. The following sections
describe the components of the framework.

2.1. Listener

JB's input components parse a stream of low-level
MusicMessages into a stream of musically useful
Phrases. This process involves two steps. First, a
ChordStream processes the MusicMessages from the
MIDInputChannel into Chords. The ChordStream
creates a new Chord for each new MusicMessage.
Since the ChordStream performs a straightforward
conversion between representations, it is reused in all
JB configurations. Second, a Listener groups the
Chords from the ChordStream into Phrases based on
the Chords' pitches and durations.

Two particular Listeners have seen the most use in
our work with the Yamaha Disklavier. A
PauseListener detects silences longer than two
seconds and begins a new Phrase after each pause,
placing each incoming Chord into exactly one Phrase.
In contrast, a FixedBufferListener stores the incoming
Chords in a fixed-size buffer and converts the buffer's
contents into a Phrase when the system demands a
Phrase to operate on. If a FixedBufferListener receives
too many Chords before producing a Phrase, it
discards the oldest Chords.

2.2. Transformers and Composers

Having parsed input from the other performers, the
system's next task is to generate its own musical
contribution. JB generates music with Composers,
Transformers, or a combination of the two.
Composers contain a musical algorithm that will
create new Phrases. Transformers transform the
output of a Listener, Composer, or another
Transformer. Varying the parameters of Composers
and Transformers can tailor them to specific musical
situations. The human performer sets these
parameters during performance by means of computer
keyboard commands or through a control panel on
the computer screen (see Figure 1). Composers and
Transformers can also derive their parameters from the
Improvvisor's PolicyDictionary, a repository for shared
information about the improvisation.

Sal Martirano's Sound and Logic was the inspiration
for the first Transformers in JB, including both

![Diagram of Improvisor Components](image)

Figure 1. Belet's on-screen buttons (left) turn several processes on and off and control the pitches used for
trills and attack harmonization, while Walker's button (right) governs the intervals of transposition, cyclic
permutations for voicing, and percentage of note omission.

ICMC Proceedings 1996 387 Walker & Belet
standard techniques, such as Retrograde and Transpose, and special-purpose techniques, such as Excerpt, a scheme for generating short excepts from an input buffer, and Voicing, his cyclical permutation scheme for revising chords [Martramo, 1971; Martramo, 1968; Martramo, 1995]. The second set of Transformers, inspired by Belet’s compositional ideas, include AttractedHarmonizer, which adds transient notes to the stack of single notes, and Trill, which turns sustained notes into trills of various intervals.

Computers create new Phrases by means of compositional algorithms. The existing Composer classes implement a variety of algorithms. HarmonyGenerator uses a library of chord voicings and rhythmic templates to create accompaniment or solo parts for a given set of chord changes. It serves as the basis for participating in jazz improvisation. RandomPhraseSource generates completely random Phrases, which are used for testing purposes. DrumPatternBuilder uses the same rhythmic templates as HarmonyGenerator to produce drum patterns with random variations. TransitionTable uses Markov chains to create material.

3. Compositional Issues

The output data generated by IB is responsive by design (as described above). At any given moment, the IB output is based on material the pianist has performed earlier. This delay creates a composition (and performance improvisation) parameter that is simultaneously a physical restriction and an aesthetic focus. This restriction is neither positive nor negative; rather, it is part of the overall context in which this music exists, and it contributes directly to the composition process and the performing attitude. “Interactive” electro-acoustic music has been the hot buzzword in our field for several years, and it therefore requires qualification for each specific use. For composition generated using IB in this context “performance interaction” is defined as a commentary by the computer on the input piano music data. It is a compositional restriction created for the specific aesthetic needs of our work, and it is not offered as the only approach for all compositional needs. The performance model of small ensemble jazz improvisation is relevant to this paradigm. As one layer of music is performed by one player, the remaining ensemble members listen to and then respond to this music in their subsequent improvisatory improvisations. Here the IB buffer is the analog to listening to the first music, and the IB algorithmic processing and output is the analog to the improvised response. This process can continue as the performer in turn listens to the IB output and then responds to it with additional improvisation, and the process can continue from performer to machine to performer.

closing the cycle of improvisational feedback. The process can be significantly expanded when two pianos are used with two computers each running IB. In our composition Cross-town Traffic the output of one performer’s IB was used as the input to the other performer’s piano, and vice versa. This creates a complex set of compositional relationships as one performer is able to respond to the music generated directly by the other performer as well as to the music being generated on both pianos by cross-related IB outputs.

A MIDI grand piano is used as the live performing instrument so that MIDI information may be used as input data for processing while the regular acoustic piano sound is heard directly. Our work has addressed composing and performing using either one or two Yamaha DXiXtra’s. The DXiXtra has been fertile ground for other composers interested in extending traditional piano performance practice and compositional models through computer interaction [Bolinger, 1992; Risset & Duyne, 1996].

As this interactive, responsive layer is governed by probabilities and the random element within the Excerpt operation, the resulting music is different for each performance on the event and phrase levels. The structural level remains constant and deterministically in agreement with the source acoustic piano music (this assumes that the performers are working with a set structure for successive performances of a given composition as that the input to IB essentially steers the output towards the intended structure). As a result, a great deal of event and phrase level diversity is achieved from performance to performance, while unity of structure and design is preserved.

4. Conclusions

Much of software development for interactive computer music follows one of two patterns. In the first pattern the programmer is also the composer. Software development is guided by the composer’s own aesthetics, and software design feedback is largely introspective. Such systems are likely to fit very closely to the composer’s conceptual model and working style, offering high productivity and usability. However, such systems rarely become sufficiently general to serve a larger audience.

In the second pattern, the composer builds a system from off-the-shelf software packages. These systems face a difficult trade-off: they either offer low-level, “aesthetically neutral” constructs and services (requiring considerable effort on the composer’s part), or they contain musical assumptions (some of which may not coincide with the composer’s vision).
IB succeeds largely because of an iterative, collaborative design process. Believing that confined, sterile laboratory conditions offer no real context for testing the usability or utility of an artifact, our goal is to get working prototypes into the hands of real users as early and as frequently as possible. By observing how these prototypes succeed or fail, we learn valuable lessons that inform the next prototype.

Our intention as composers and performers is to explore and augment the ensemble improvisation process. Using IB has proved to be an aesthetically successful means of pursuing this goal.

References


