MusicTalk: A Tool Driven Approach to Computer Aided Composition

Dr. Andreas Mahling
Staatliche Hochschule für Musik und Darstellende Kunst Stuttgart
Urbanstraße 25, 70182 Stuttgart
Germany
email: andreas_mahling@hotmail.com

Abstract
This paper describes MusicTalk, a system for computer aided composition. It is based on earlier scientific work on how adequate representations of musical knowledge can aid in shifting the complexity barrier in music software [BöckerMahling1988, Böcker et al. 1990, Mahling1991]. Unlike MusicTalk's predecessors, this time, the development objective has been the implementation of a workbench for exploring musical structure which is flexible enough to integrate new compositional methods without the need for a complete reimplementation at a later time. This flexibility is provided by a tool-driven approach. Composing with MusicTalk is done by choosing tools having private user interfaces from a tool library, supplying input to their parameters while the rehearsal loop is active and storing satisfying results in a material library or arranging them in a simple sequencer. The library is not just a container for the final composition but may store complete evolution cycles of compositional alternatives. MIDI is generated for sound output but extra support for specific sound computation programs like Csound is also provided.

Introduction
Music software with scripting abilities or embedded programming languages usually falls short of providing appropriate development environments. This makes it a cumbersome task to add extensions, and sometimes prevents this entirely. Even systems based on powerful programming languages like LISP, usually do not provide a complete development environment but require the user to buy it as an option. Second, although approaches which extend a general purpose programming language with musical constructs provide the most flexible way for enhancing a systems abilities, they usually do not have adequate help on how to use all these constructs (a Windows help file is not considered to be sufficient by the author). Last but not least, a reasonable set of functionality for computer aided composition systems like maintenance of compositional alternatives (including documentation!), draft arranging, rehearsal or interfacing with externals, stay the same, independent from the specific compositional approach taken by some user. Therefore, the objectives for the development of MusicTalk, which started in late 1996, were 1a) to provide a frame equipped with functionality to execute compositional-work-everyday-tasks, 1b) to equip this frame with an interface that allows for isolating everyday task functionality from compositional procedures, that way facilitating integration of new compositional tools, 2) implementation of, among others, a tool for that frame providing musical extensions to a programming language with the possibility to assemble language expressions in a totally dialog-driven manner, this way reducing the number of lexical and semantical mistakes which especially burdens new users of a system and 3) to embed all this in a programming language with a powerful integrated development environment, available to musicians without any charge.

MusicTalk Overview
MusicTalk is written in Smalltalk-80 and is completely integrated in Smalltalks development environment: VisualWorks 2.52, originally marketed by ParcPlace Systems, a former spinoff of the Smalltalk-80 Research Group at Xerox PARC. Past experience has shown that Smalltalk is adequate for handling complex musical tasks [Krasner1980, Pope1992, Mahling1993, ScalalettiHebel1997]. MusicTalk can be used for free if used for musical purposes only and can be downloaded together with extensive documentation from http://www.s.netic.de/mahling/musicstalk.html. Since 1998 it is used in my courses at the music conservatory at Stuttgart and several modules have been adopted by colleagues for their work.
MusicTalk is not bound to a specific compositional technique, but instead offers a workbench for tools of which each can have its own user interface and preset library. MusicTalk does not attempt to do everything in real time. The degree of interactivity of its rehearsal loop depends on the complexity of the algorithms used. It works on the structural level of music composition and provides interfaces to sound computation programs and devices.

MusicTalk can be extended either by using means provided by tools, which allow user-extensions without programming, or by directly writing Smalltalk code. Tools, which allow for storing parameter settings (presets) can have preset dependent documentation to be added by the user. For this purpose, MusicTalk generates HTML-Templates on demand as a starting point. This increases consistency between “development” state and documentation of a project, because implementation is referenced for template generation. Furthermore, it helps new users to find their way to the system and keeps experienced users on the right track in large projects.

![Fig.1: The MusicTalk Workbench Window](image)

A MusicTalk song or project stores general settings for playback, e.g., MIDI time resolution, properties and arrangement of patterns (the track sheet), tool specific user definitions (if applicable), presets and the contents of its material library. Projects can also import individual portions of data from other projects on demand. Furthermore, MusicTalk is maintaining documentation consistency of user projects, e.g., moving user defined objects between project folders will also move associated documentation.

Fig. 1 shows MusicTalk’s workbench window, i.e., the systems main window. Together with the menu bar, the Track Properties and Track Sheet panes provide basic sequencing functionality to avoid too much program switching when preparing a draft of the pieces arrangement or for doing quick rehearsal. The Tool Library pane displays a list of available tools for data manipulation, organized in categories, and selectable by the user. The Tool Parameter View presents the tools interface, which in turn is used for supplying input to the active tools parameters. When parameter specification for a selected tool has been finished, the Rehearsal Loop Control Panels interface widgets can be used to apply the tool, listen to the result, apply the tool again or change parameters in between if output is not satisfying, and if result is satisfying store the result in the material library. The material library is a pool of all project relevant data. Its content is displayed in the Material Library pane, which shows material, grouped into categories. Data can be stored without the need for arranging it first, like a sequencer would demand. The material library is also meant to be used for maintaining alternative solutions to a compositional problem and to look at a solution from different viewpoints. The material library can be manually filled with entries step by step or several entries can be added automatically in one pass through some composition module. The library also serves as a blackboard for compositional algorithms whether built-in or user-defined. Finally, an arrangement (draft) can be assembled from individual entries in the material library by dragging them to the Track Sheet window.

**Architecture**

MusicTalk cannot be equated with its workbench window. Moreover, its a huge collection of musical objects and behaviors embedded in the integrated development environment (IDE) of Smalltalk. The current version (MusicTalk 1.42) consists of more than 200 classes with more than 4,600 methods. The basis for almost each of its tools is its musical knowledge base, a collection of objects which capture common musical knowledge in a procedural form to aid in representing musical data in a software system e.g., objects for MIDI messages, (untempered) Pitch, Scale, Meter etc.

![Fig.2: MusicTalk’s main components and interfaces.](image)

Although the Smalltalk-IDE is the most flexible means for extending MusicTalk’s abilities, existence of profound programming or Smalltalk knowledge on the user’s side cannot be assumed in general. Therefore, MusicTalk adds a couple of user interfaces to the Smalltalk IDE, which allow adding
functionality without Smalltalk knowledge. It is assumed here that readers are familiar with what a sequencer is used for so that this chapter can concentrate on a more special UI as an example: The EventGenerator is a tool which was influenced by the stream principles introduced by Rick Taube's Common Music [Taube1991] but adds a type system, a generic dialog interface and, last but not least, a lot of new stream types. Through this interface the user is assembling stream specifications by selecting "language constructs", e.g. streams, and input values, e.g. a duration, from dialogs. Dialogs and their content are context sensitive and the type system is used to constrain stream input, e.g. one of the basic stream types is a HeapItemStream, which allows random serialization of an element set. No prior restriction is made to the kind of elements for this stream, however if it is connected to the duration input of a MIDIEventStream (which is computing MIDIEventNotes) the dialog for the elements: parameter of the HeapItemStream automatically presents to the user duration compatible value types only.

Beyond MusicTalk's workbench window there are tools that either help in developing and maintaining new system modules or support quick & easy input or editing. E.g. when developing new program modules which make use of Windows multimedia communication interface (MCI), it might be helpful, for debugging purposes, to have some tool for selective evaluation of multimedia expressions. This is what the MCIScratchPad is meant to be used for. For scribbling music ideas or just for parameter input for composition tools, e.g. supplying a pitch scale via an external device, the MIDIScratchPad might be appropriate, which provides direct access to all MIDI devices available to a specific hardware platform. All these tools are intentionally kept simple and easy in handling because one of the main goals of MusicTalk is not to reinvent the wheel, but to integrate "foreign" software whenever it seems to be more convenient. Therefore MusicTalk does not provide tools for fancy MIDI editing or conventional music notation but offers a mechanism to call external programs in a synchronized or unsynchronized way instead. For audio output, MusicTalk is not just calling the external program, CSound in this case, but is also integrating associated tools, e.g. for computing phase vocoder analysis files, and provides a parsing feature that allows maintenance and selection of CSound instruments from within MusicTalk's TrackProperties pane as if they would be patches of a conventional MIDI synth.

Exchanging data with third-party programs is currently supported via MIDI standard files (MSF) and MusicTalk's version 1.5 will include an internal documentation tool (DocBrowser) which integrates user input with system generated parts derived from implementation or user presets in an explicit way. An external HTML editor can be used for writing new documentation chapters or to tune system generated documentation templates. Another objective of this tool is that output format can be adapted easily to future needs which might be caused by new standards and interaction possibilities.

Workbench Tools

Except for its sequencer subpane, MusicTalk's workbench window is entirely driven by its tools, i.e. what can be done and what not depends on the tools provided by the workbench windows Tool Library. (In this chapter the term tools is used for tools which can be selected from within the Tool Library pane of MusicTalk's workbench window.) Each such tool can have its own user interface and, if appropriate, a means for storing selected settings of parameters (presets). Tools themselves can be selected from the Tool Library pane (see Fig.1) which classifies tools according to given categories. So far, new tools can be added by writing Smalltalk code only. Tools may support any compositional technique and can provide any suitable user interface. As an example of how different tools can look like, three examples will be introduced here: A simple tool for copying the rhythmic structure of one pattern to some other, a more elaborated tool for generating drum patterns and a complex tool for building event list generators.

The Rhythmicize Tool

The Rhythmicize Tool, which is rather simple, replaces the rhythmic structure of one phrase (Melody Track) with that of some other one (Rhythm Track). If the melody track lasts longer than the rhythm track, the rhythm track is repeated as long as notes are needed by melody track for rhythmicization. If rhythm track lasts longer than melody track, superfluous notes are ignored. Rhythmicization is finished when all notes of melody track have been processed.

Fig.3: Applying a simple tool for (re)rhythmicizing.

Fig. 3 shows the user interface for the Rhythmicize Tool: two input fields for the parameters of the tool onto which MIDI phrases from the material library can be dragged with the mouse. The tool is applied by pressing the Apply button and the result can be listened to by pressing the Listen button. This process might be repeated until output is considered to be worth to be remembered. This is especially useful for tools using random procedures. Each new run of the tool overwrites the output of the previous run. The output of the most recent run of the
selected tool can be stored in the library by supplying a pattern name and pressing the Accept button. Because of its simplicity, the Rhythmicize tool does not provide the ability to store presets.

The Drum Pattern Generator

The Drum Pattern Generator tool generates drum patterns from sequences of likelihoods supplied for each drum instrument separately by the user. This idea is not really new. Similar approaches for instance can be found in systems like KeyKit [http://thompsonresidence.com/keykit/]. This tool is presented here to show how different tools can be integrated in MusicTalk's workbench window.

Fig. 4: The Drum Pattern Generator

Fig. 4 shows a user defined Drum Pattern Generator preset, providing likelihood value sequences for bassdrum, three toms, high-hat and cymbal. Each field in the likelihood matrix specifies the likelihood for a hit of some drum instrument on a specific fraction of a beat/measure. E.g., value sequence for the bassdrum defines that a hit happens on the 1st, 5th, 9th and 13th sixteenth note of a measure, assumed that one row of the matrix represents 4 crotches (each divided into 4 sixteenth notes). The high-hat specifies a hit on each sixteenth note except the last one, when "closed", and a hit on the 16th sixteenth note, when "open". However, unlike for the bassdrum, a hit is not 100% sure but happens with a likelihood of 50% for odd sixteenth note counts and 25% for even sixteenth note counts only (except for the open high-hat hit, which also occurs with a likelihood of 50%). This means, that each run of the matrix will usually output a different high-hat pattern. This does not mean that instrument patterns need to be completely arbitrary. For instance, likelihood sequences for the toms are given in a way that, although individual tom hits may vary, rolls will always move from the high-mid tom to the low tom within a single measure.

The Event Generator

The Event Generator is not just a single tool but a tool construction kit. It is based on the idea of computing output by interconnecting streams, each representing a different algorithm. There are streams for random selection principles, arithmetic, event generation, envelope generation, transition networks (graphs), manipulation of entire phrases like morphing, parameter mapping, filtering, arranging, storing and retrieving data and more. The EventGenerator is used by assembling a stream specification which, when finished, can be applied to compute (music) output. Stream specifications can be stored as presets for later reuse and may declare their own parameters. Because the assembly of streams is entirely dialog-driven, new stream types can be defined by the user without programming. There is no way to supply syntactical or semantical bad input. Assembled stream specifications will always run, however, they might not always do exactly what the user intended to do initially.

Fig. 5: Sample stream specification for the Event Generator.

Fig. 5 partly displays a stream specification using a user-defined stream called TimeScaledSyllableEvent. This stream possesses four arguments: syllable, key, attackVelocity and duration (in the fig., first two of them can be seen only). To differ between system built-in and user-defined streams, user defined streams are enclosed in braces. This stream generates events which control a program running on an external device (a Capybara-320 from SymbolicSound [http://www.symbolicsound.com]) via MIDI for resynthesizing vocal phrases. The stream allows for selective playing of individual syllables out of a continuous wavesample with time variant time-stretching and pitching. To facilitate experimentation, values for syllable start times, event generation for syllable selection etc. have been hidden in the definition of the TimeScaledSyllableEvent stream, i.e. syllables can be used like abstract atomic entities ("first class objects"). Actually a new Event Generator user type Syllable was defined to associate reproduction relevant data with each syllable instance. The MTRandomItemStream in line 3 generates random sequences of syllables and the MTRandomStream in line 7 computes pitch values to be used for each corresponding syllable. Pitches are computed by randomly moving up or down in thirds (line 11-13) starting with pitch value g3 (line 9). The duration of
each syllable, in this case, is also determined randomly. (This stream cannot be seen in fig. 5).

There are still more things to tell about the features of this tool and how to use them. Interested readers may want to have a look at my website mentioned above. Those who do not want to download the entire MusicTalk setup (approx. 10 MB) before further reading may want to download a standalone version of MusicTalk's documentation first. Although derived from an older MusicTalk release it already contains in-depth information about the Event Generator.

Future Development

Most of the enhancements planned for future MusicTalk releases are related to the Event Generator tool, because it was shown to be flexible, easily extensible and, in the meantime, reached a state so that it could be used by students for medium sized projects like automatic modulation under consideration of voice leading within one semester. Among such enhancements is an editor for specifying transition nets graphically and a postponed computation of the MIDI layer, i.e. abstract objects will also be usable in both, the MusicTalk's sequencer and material library. Extending the system set of workbench tools, without programming in Smalltalk, is another add-on coming up in the near future which is outlined in the next paragraph.

A main development target for MusicTalk was to have a "good" feedback loop when exploring musical structures and its current Rehearsal Loop seems to be a step forward towards that goal. It is fast, it can be used on any abstraction level and intermediate results can be maintained easily. However, during teaching lessons it came out that an extra user interface assembled just from widgets for the compositional parameters would facilitate work even more. Currently a user is still faced with a stream specification on this final level, i.e. a stream spec representing the process for computing the entire composition or at least a part of it. Such a specification, of course, shows more detail than is required for controlling it. Furthermore supplying or changing a parameter value currently needs more interaction steps because of its dialog-oriented nature. Simply dragging a widget like a slider would vastly increase input speed. Additionally, arrangement of parameters could be discarded from demands caused by lexical requirements for specifying a stream and could just be guided by what is most convenient for controlling the compositional process. Such a user interface would urge two further enhancements: an automatic looping feature for the rehearsal loop and a stream-to-tool-converter module for the workbench window. The automatic looping feature would automate the apply-listen cycle, i.e. MusicTalk would always replay the results of the most recent application of a stream specification and changing a widgets value on the user interface would automatically update the result of executing the stream specification with the changed parameter values. Changes would start to be played immediately upon beginning of the next loop cycle. With the availability of a new user interface the introduction of a stream-to-tool-converter is obvious, because all that is needed to add a new tool to the workbench is to supply an executable algorithm and a user interface for controlling its input parameters.

Conclusion

MusicTalk's development did not start from scratch but benefitted from three predecessors, first of which started development in 1986. Over the recent 5 years MusicTalk was verified to be a stable platform for compositional work and future extensions. It is based on one of the most powerful development environments for programming, provides a flexible way for assembling compositional processes in a dialog-oriented manner which does not require Smalltalk knowledge, provides a workbench window supporting draft arranging, immediate rehearsal, interfacing to MIDI devices and formats, integration of external editing programs and modules for transparent maintenance of an entire composition process, functionality which is always needed, independent from a specific compositional approach. As a result, MusicTalk is raising the starting point for implementing new composition principles to a higher level and reduces the amount of time needed to get them to run.

References


