

QSketcher:

An Environment for Composing Music for Film

Steven Abrams, Ralph Bellofatto, Robert Fuhrer, Daniel Oppenheim, James Wright

{ sabrams, ralphbel, rfuhrer, music, jimwr
@us.ibm.com }

Computer Music Center
T. J. Watson Research Center, IBM
P.O. Box 218, Yorktown Heights, NY 10598
www.research.ibm.com

Richard Boulanger, Neil Leonard, David Mash, Michael Rendish, Joe Smith

{ rboulanger, nleonard, dmash, mrendish, jsmith2
@berklee.edu }

Berklee College of Music
1140 Boylston Street
Boston, MA 02215-3693
www.berklee.edu

Abstract

We describe QSketcher, a new environment for composing music for film. The main design focus is the support of the early stages of the creative workflow, from idea conception through realization, rather than the mere order and synchronization of musical fragments with film. This paper describes the design process and rationale, the system, the user environment, and how they relate to one another. Novel aspects of the system include a free-form 'idea space', a main workspace that can be configured to individual needs, an "idea capturing" facility, a workflow tracking mechanism through which previous workspace states can be examined and restored, and the ability to create a variety of relationships among musical elements.

Keywords: creativity, composition, user interface, interactive, music representation

1. Introduction

The music composition process has evolved over millennia as a balance of opposites: inspiration versus perspiration, broad formal approaches versus minute detailed work, macro-level (or structural) conceptualization versus micro-level (e.g., note-level) editing, and so on. When composing in the traditional ways, one moves frequently from mode to mode: an idea pops up, one captures it, and then thinks critically about and develops it. Another idea pops up, and the process iterates. One can easily shift focus to thinking about the entire composition and how a small motif relates to the whole, for example.

Unfortunately, the technological tools available for composing music do not readily support this kind of creative workflow – they are geared towards realization of preconceived ideas. Thus, the user interface found in many commercial applications models recording studio machinery, namely the multi-track recorder and mixer. This model has little to do with the traditional process of creating music. So, although the multi-track recorder metaphor is appropriate for the mixing and audio post-production stages,

it provides little leverage in capturing and developing musical *ideas*, which lie at the heart of the early stages of the creative process.

In addition, the environments for composing music that have evolved over five decades in academic and commercial settings still fall short in their ability to manipulate music directly in terms of musical concepts. Computer languages for composing music, for example, typically model limited structural aspects of music through a score, e.g. instrument, part and score, and provide a suite of algorithms to manipulate that score. The score is often expressed in physical terms, such as frequency and amplitude, and the tools provided for its manipulation are often borrowed from computer science with few changes. This detachment from the intuitive musical concepts in which composers think, and from the musical experience itself, places an unnecessary cognitive burden on the composer. Computers are supposed to make work easier, not harder!

Our goal was to develop a tool that overcomes these limitations. To focus our efforts, we addressed a specific musical task: that of scoring film. We felt that this would not overly restrict the musical style, that general issues relating to supporting computer-assisted composition would still need to be addressed, and that the film environment in particular provides structure that resonates with many of our ideas. We formed a cross-disciplinary research team consisting of computer music researchers, system developers, and composers (note that all team members have strong musical backgrounds; most have strong technical backgrounds). Together, we examined the creative workflow, critiqued existing solutions, and focused on areas that we felt needed attention. We also employed a highly iterative development strategy with our composers involved in most stages. Our focus on the creative workflow and musical concepts had a profound impact on the system design, from the underlying data structures used for music representation to the overall user environment.

The result is *QSketcher*, an environment designed to allow composers to move fluidly between dichotomous

modes (inspiration/perspiration, capture/manipulate, and macro/micro editing levels), while directly supporting a variety of common compositional concepts, so that composers can work using the terms in which they think.

Interestingly, these dichotomous modes are also present in a wide variety of creative tasks – writing, drawing, preparing presentations, as well as more technically-oriented activities such as software design and development, architecture, and even the act of research itself. The concepts uncovered during the QSketcher project are important in these domains, and many of the solutions implemented would prove quite useful there. Several of the authors are currently adapting these techniques to other areas. In fact, we believe that the “tip of the iceberg” of an important research area in HCI is emerging: developing information technology tools to support creative work.

The remainder of this paper describes creative workflow as we understand it, the demands that supporting creative workflow places on a system, and the specific aspects of QSketcher developed to address these demands. We will draw parallels to other domains where appropriate, and conclude with remarks on future directions for this research.

2. Creative Workflow: Capture, Organize, Manipulate

Throughout our dialogue, three general areas emerged in which composers felt the need for better tool support: the ability to quickly *capture* musical ideas, *organize* those ideas in a useful manner, and *manipulate* them in musically meaningful ways.

It was clear that the composer's workflow would be much more fluid if the system made it trivially easy to input their ideas - as graphical sketches or scribbles, textual annotations, music played on a keyboard, and so on. Any break in the flow — to enter record mode or handle other technicalities of operating the system — could disrupt their creativity, and an idea might be lost. Ideally, every idea should be captured, and capturing should be allowed at any stage in the creative process. To support this, we incorporated features such as the “Infinite TakeVault” and an integrated freehand drawing tool, both described later.

With every musical idea captured, the system must also prevent information overload. This can be accomplished by providing intuitive and powerful mechanisms for organizing ideas, relating ideas to the relevant music or film content, and searching for ideas. QSketcher supports these goals with features such as an integrated content palette (a database allowing for searching materials by many criteria), and the ability to organize ideas alongside musical materials using the visual layout of windows.

Finally, the system should provide musically meaningful ways to manipulate content, thereby allowing the user to rapidly explore the musical space, experiment with ideas, and develop and structure musical fragments into a final work. To this end, QSketcher supports high-level structural

manipulations as well as precise low-level editing.

As we proceeded with the design, we uncovered four pervasive higher-level issues that have great impact on the system's effectiveness in each of the above areas: *conceptual orientation*, *context*, *state*, and *relationships*.

Conceptual orientation refers to the system's use of musically meaningful concepts in its visualizations and controls. Composers think about music using musical concepts such as motif and development, crescendo, transformation, or tension and relaxation. As an example, QSketcher allows the composer to structure his composition in an arbitrary manner – as a hierarchy consisting of notes, phrases, sections, and cues, as an arrangement of tracks, or any combination thereof – and then to manipulate the composition using that structure. Another example is QSketcher's ability to adjust tempi by directly manipulating an object representing the length of a structural element, in relation to film events in the global timeline.

Context relates to the composer's mental state while working on a particular problem. For example, composers often cover their desk or walls with sketches of musical ideas, outlines of musical sections, fragments of music and motifs, photographs of objects relating to their work, notes about intent, scripts, cue lists, todo notes, napkin scribbles and the like. These objects help the composer mentally work out various relationships and musical processes that are an important part of musical creativity. In previous systems, the computer and monitor hold only a small portion of the working environment. We believe it is crucial to capture much more of the working environment “inside the computer.” Then the computer can do far more: track relationships between materials and tasks; capture ideas along with the rich context in which they were created; support a much smoother flow between different tasks; and quickly recreate different versions of the environment according to the needs at hand.

While we do not expect to model the cognitive facets of these relationships directly, we recognize that the visual layout of objects in the composer's workspace often reflects important aspects of his thought process. We therefore developed a user interface that allows him to place *anything anywhere*, creating what we term a *visual layout*. To assist in recovering the mental context as it relates to a given problem, layouts are remembered and easily recalled within QSketcher. These ideas were central to the design of the “*Idea Space*” (see figure 2), described in a later section.

While *context* loosely relates to a set of things that the composer associates in his mind with a given problem, *state* is connected with both the entire composition and the musical experience at a given point in the work. When working on a given section, it is important for the composer to correctly assess what the musical experience (state) will be. Listening to the work from the beginning to that point in time is time consuming, and often disrupts the workflow. Moreover, the preceding sections may be incomplete.

Our design addressed these issues in several ways. First,

the visual environment presents both a high level view of the entire composition and its structure (the “*Project Space*”), alongside the specific locale that is being worked on (the “*Idea Space*”). Second, the Idea Space can be arranged to display the kind of views and amount of detail that provides the most appropriate feedback on the musical state. Third, sketches and placeholders can be inserted even where no music yet exists, to visually show musical intent.

The notion of *relationships* arose because composers mentally relate various materials within a composition in a number of ways. The composers thought it would be useful to visualize these relationships so that one could view the musical materials in relationship to some compositional process, and not only by the order in which they appear in the composition timeline. For example, composers often use motifs – recurring thematic elements. In most tools, the composer can copy a motif from one place and re-use it in another with a “linking” operation. However, if, as often happens, the composer alters the motif in some particular context, the link – and, in most tools, the relationship – is broken. Of course, these elements *are* related in the composer’s mind: they are instances of the same motif.

To address this problem, our music representation provides “ancestry links” that point from the copy back to the original. When a musical fragment is copied and pasted, an ancestry link is created. The composer may adjust each motif instance to the surrounding musical context and the system retains this link, even if the music ultimately differs substantially. The composer can later query the system to see all recurrences of a given motif. (see Figure 6, Figure 8).

It soon became apparent that there are many other kinds of relationships that would be useful to establish, both *episodic* and *semantic* (functional). An *episodic* relationship associates an object with a certain concurrent or coincident activity. For example, a composer may not remember in what folder a certain melody patch is stored, but may be able to recall what scene of the film was showing when he first improvised it. The scene is thus episodically related to the melody. A *semantic* or *functional* relationship connects objects that relate by a certain function, such as an expressive curve and a musical phrase, and its presence directly affects the final result. *Containment* relationships create a hierarchical structure, such as a violin melody within the string section in the second movement of the piece. A *Process* relationship indicates a sequence of actions that was used to shape musical material, such as a transposition, filtering, or inversion. *Referential* relationships indicate, for example, the places where a musical entity is used, such as all the appearances of a given motive. In addition, the composer may want to create her own *categorization* relationships for grouping different objects together in a way the composer finds meaningful.

The above ideas impacted our design in that the underlying music representation had to provide for establishing relationships without their being functional, and the user interface had to provide visualization and

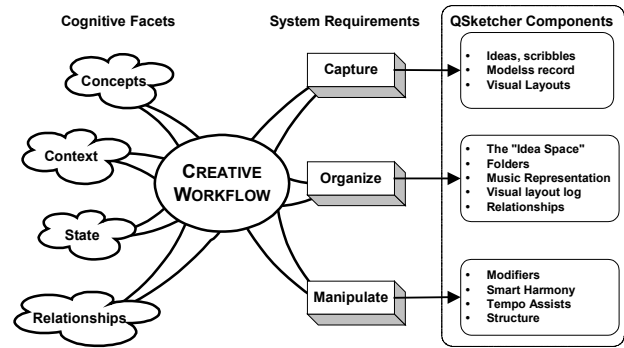


Figure 1. Overall cognitive aspects and system requirements, and related QSketcher components

navigation facilities for managing them. Note: well after our basic design was established, we discovered that our taxonomy of relationships was similar to classic cognitive models of memory organization. (Tulving 1972) first distinguished between episodic and semantic memory, and later (Tulving 1987) added procedural memory as an additional class (along with propositional memory, which encompasses episodic and semantic memory). Our notion of episodic relationships corresponds to episodic memory; our Process relationship corresponds to procedural memory; other relationship categories all relate to semantic memory.

The diagram in Figure 1 summarizes many of the points discussed above and identifies some relationships between the requirements of creativity and system components. It is important to note that in reality the relationships are more complex than indicated in the diagram – most system components relate to more than one workflow stage or cognitive facet (Oppenheim 1991). The following sections will discuss the system from the perspective of the workflow: capture, organize, manipulate, and context. We will end with a discussion of the important aspects of the music representation that reflect and support these ideas.

3. The Composition Environment

The design of our environment is rooted in several simple observations:

- Balancing opposites is a way of life for creators: inspiration vs. perspiration, top-level structure and form vs. minute details, sketching vs. refining.
- Creators have many different work-styles: no single approach or process is sufficient. We need to support “structured noodling” as well as formal construction
- The workplaces of creative people are generally littered with meaningful arrangements of “stuff.”

To accommodate widely varying work-styles, the tools and environment should be both flexible and easy to customize. This is often much more important than the sheer power of each tool: if they are hard to use (or discover, or access when you want them), power tools give little benefit. And it is important to have tools that work at each of the various levels – fine editing tools as well as gross or structural manipulation tools.

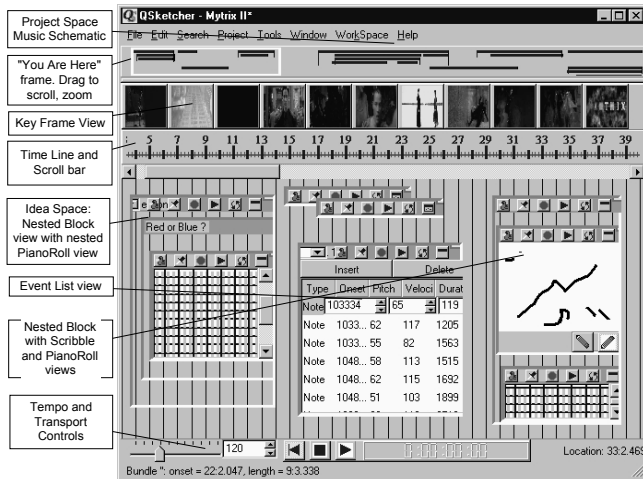


Figure 2. The Idea Space and Project Space

Our environment is divided into roughly three parts: the *Project Space*, the *Idea Space* and the *Database Palette*. The *Project Space* manages project-level information, activities and navigation. This includes foldable schematics of the music and visual structure (which can be very different), as well as more traditional high-level film context (overall timeline, important time markers, video ‘key frames’). The *Idea Space* provides the project’s main ‘work surface’. The Idea Space embodies a ‘boundless sheet of paper’ metaphor, in which time runs from left to right. Content of all forms (music, post-it notes, scribbles, control curves, compound assemblies...) is captured, displayed and manipulated on this work surface, in a user-defined arrangement of nested views. Some of this content is playable; other content is simply presented for reference. The Idea Space is also used for managing the project’s hierarchical structure. Three kinds of view modes (*Embedded*, *Pinned* and *Floating*) clearly distinguish between project content seen ‘in time’, context-sensitive expanded and referential views of content, and free-standing tools and palettes. We discuss this further below

The *Database Palette* holds all kinds of material: raw materials and finished sections; quick sketches – potentially any system object including cue sheets and phone logs, tool configurations and visual layouts. The Database Palette includes facilities for browsing or searching through content, and is described in more detail in section 5.

QSketcher provides several ways of presenting musical and music-related content. You can view a musical entity as a block-oriented structural view, as a ‘piano roll’ event display, as a textual event list, as expressive curves (e.g., tempo, pitch, volume), or even as sketches and textual annotations. Any or all of these views may be displayed simultaneously. QSketcher affords the user considerable flexibility in arranging these views so as to display exactly what the composer needs to see, where and when he or she wants to see it. For example, most views can occupy very small amounts of screen real estate, by adjusting the amount of information displayed, facilitating the transition from

macro to micro level work.

QSketcher is designed to help the composer visualize their mental context by several means. The Project Space visually presents the global compositional structure as a compact ‘Music Schematic,’ showing the top 1 to 3 levels of the content hierarchy. A separate ‘Film Schematic’ presents the global structure of the film, showing the structural counterpoint between film and score. A highlighted rectangle (the ‘You-Are-Here’ view) on the schematics shows the portion of the composition currently visible in the Idea Space. This rectangle can be manipulated directly to effect scrolling and zooming.

QSketcher provides a novel mechanism for viewing content in or out of temporal context. Normally, a content view appears within the Idea Space so that the left side of the view rectangle indicates the content’s onset, while the rectangle’s width indicates the content’s duration. We call this an *embedded view*. Moving the view rectangle changes the onset of the content; resizing the rectangle changes the content duration (by manipulating tempo, changing a loop or repeat factor, or perhaps clipping start or end of the content). In short, changes to embedded views directly affect the composition (and appear immediately in the Music Schematic). Most importantly, the contents of an embedded view are tied to the indicated point on the film timeline, and are rendered at the specified time relative to the containing section.

However, it is often desirable to view a piece of music content out of context, e.g., to examine a section near the start of the composition while working on similar material in another section, or to expand a view for detailed editing purposes. For such purposes, we provide *pinned views*. A pinned view is in effect a duplicate view that can be arbitrarily resized or moved without affecting the temporal properties of the corresponding content. The term ‘pinned’ derives from the fact that these views are attached (pinned) to a specific time location in the composition. Thus, scrolling the Idea Space causes the pinned view to move, disappear and reappear just like embedded views. Unlike embedded views, pinned views are *not* rendered when the containing section is played. Pinned views act as if you literally pinned a copy of some content onto the score, and may be collapsed to conserve space if desired. Composers may use a pinned view to show a relationship between two pieces of content – and as another reminder of context.

Finally, QSketcher provides *floating views*. A floating view is like a pinned view in several ways: it may be collapsed as needed, modifying its geometry has no temporal consequences, and it does not get rendered when the composition is played. However, unlike pinned views, a floating view is not anchored to a particular project location, but rather to a screen position. Thus, scrolling the Idea Space has no effect on floating views; they behave as if literally pinned onto the screen. Since views can hold tools as well as musical content, this provides a very flexible means for organizing the workspace.

The compositional process frequently requires shifting from one focus to another. For example, one may start by working on the melody, which then leads one to change the harmonic progression, which alters the harmonic rhythm, which in turn leads one to consider alternative rhythmic skeletons, and so on. Each of these activities might be best carried out with a different arrangement of views and tools.

Our system addresses this need by providing simple ways for capturing and recalling a snapshot (configuration) of the set of views, including tool configurations and the portion of the composition (or the database) that they were viewing. One can have many such snapshots available at any time for recall at the click of a button. Unlike other systems, it is not necessary to explicitly save a snapshot. QSketcher automatically captures a snapshot whenever it detects that significant changes have occurred. (A short time delay is used to avoid capturing many almost-identical snapshots.) We call this the “You Were There” feature.

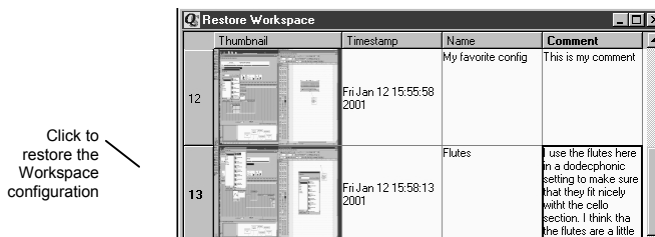


Figure 3. The Workspace configuration Log

The snapshots (whether manually or automatically captured) appear within a Workspace Configuration Log; each snapshot is represented by a thumbnail graphic of the screen layout along with the wall-clock time and project time position when it was captured. Comment and title fields are provided for optional annotations by composers. The log may be sorted by any field; we hope to provide further sorting and retrieval mechanisms in future.

4. Capture

Ideas come in many forms, at any time, particularly in the early stages of a composition. Moreover, their usefulness is often not obvious a priori, but rather, becomes evident only later. Unfortunately, ideas are also evanescent, disappearing if you take too long to capture them. As (Stravinsky 1942) put it, “In the course of my labors I suddenly stumble upon something unexpected. This unexpected element strikes me. I make a note of it. At the proper time I put it to profitable use.”

Therefore, we adopted the following design principles:

- Nothing should ever be lost: all captured content should go into an “Infinite Take Vault.” Content should be annotated automatically, to preserve its original context. Disk storage is cheap; good ideas are priceless.
- The system must support many kinds of content, with modeless or near-modeless capture
- Facilitate concurrent user activities and “constructive noodling” (unplanned improvisation).

QSketcher’s capture mechanisms are novel in two ways. First, we recognize that hand sketches, graphics clips (e.g. still frames extracted from video), audio clips (e.g. music or recorded conversations), textual annotations, musical fragments and skeletal compositional elements (e.g. A-B-A structures), all participate in the creative process. QSketcher can capture each of these forms of ideas, and allows the user to place them in the Idea Space, serving many possible uses:

- fragments, improvisations, or musical sketches leading to the final work
- visual cues of compositional intent
- placeholders for future work
- several perspectives on the same musical entity

Second, our capture mechanisms provide a fluidity that few environments offer, by virtue of both the “boundless paper” metaphor and their nearly modeless behavior. Thus, the user can watch the film while sketching in the Idea Space, playing on the MIDI keyboard, typing notes, whatever is appropriate. The system captures the ideas, along with relevant contextual information. For example, to record from a MIDI keyboard, one simply begins playing without explicitly entering ‘record mode’; the system actively listens to the MIDI ports at all times and records everything played. The recorded material is stored in the database’s “Take Vault” folder, and annotated with creation attributes including the wall clock time and project location. If the Idea Space has an active insert locus, the new material is also inserted as a new block at that location.

By extension, to sketch a tempo curve, one should simply be able to pick up the mouse and begin drawing on a block’s background; to create a post-it note, one should simply begin typing. Clearly, this could introduce ambiguity in interpreting gestures. Clicking and dragging the mouse could signify that the user wants to draw a shape, create a selection, or move an object. The ambiguity might be resolved by the choice of input device, as an extension of the above paradigm, which uses the MIDI keyboard exclusively as a recording device. For example, a graphics tablet could be reserved for freehand drawing and gestural control (Wright et al 1997); the mouse, for selection and direct manipulation. More work is needed in this area.

5. Organize

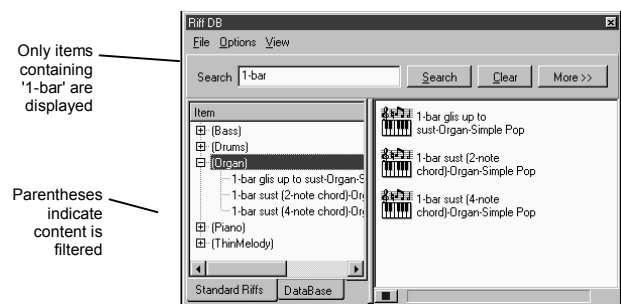


Figure 4. The Database Palette(Query Mode)

Creative workers often suffer from “content overload.” Working with a system over an extended period of months or years, or with large numbers of samples or other canned materials, it often becomes exceedingly difficult to locate the right content to use as raw material for another composition. To help combat this problem, QSketcher was designed with an integrated “database” that can house any kind of content: entire compositions, melodic fragments, chord progressions, and so on.¹ This is possible in part because the database is built using the same kinds of objects that the system uses to build compositions.

To make the database search mechanism even more useful (while avoiding undue burden on the composer), QSketcher automatically captures contextual information as objects are created or modified. This includes wall clock time, project timeline location, containment hierarchy, ancestry (from which an object derives), etc.

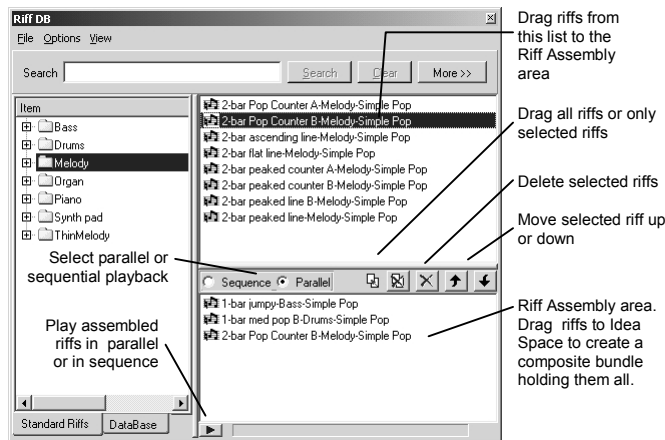


Figure 5. Database (DB) Palette: Riff Assembly area

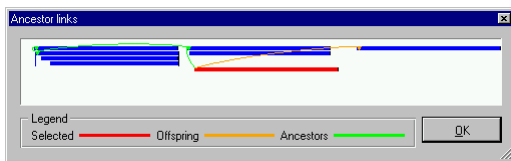


Figure 6. Ancestry Links Display

The database provides several basic functions. First, it allows you to find anything stored in the system (even outside the current composition) by qualitative queries on attributes. The mechanism supports searches on:

- creation or modification date/time (“last Thursday”)
- name of the musical fragment

Slated for future implementation are searches on many other attributes, including:

- arbitrary symbolic tags (“happy”)
- musical attributes, such as key, tempo, instrumentation, melodic fragments, chord sequences

¹ In fact, the database can theoretically house rhythmic skeletons, tool configurations, and view configurations, but we have not yet exposed this ability to the user.

- where it was used (in what compositions and where within)
- the portion of the film that was displayed while the riff was originally captured

Search results can be displayed either as a flat list, or in hierarchical context. The database was designed to allow the composer to create his own hierarchical organizations of material that exist alongside the system-defined ones. For example, one can create a database folder called “My Cool Bass Riffs” and place music blocks into that folder. The user can also create multiple database views, each displaying a different portion of the database. Each view can navigate and browse the database, or the results of any search.

The database palette naturally provides a means for auditioning the content, but adds a unique “assembly” area allowing the composer to experiment with arrangements, grooves, counterpoints, etc. and drag the result off into the composition (see Figure 5).

6. Manipulate

The need to manipulate musical materials is obvious, and many tools exist for this purpose, such as piano roll editors, event list editors and the like. While these are certainly useful (and QSketcher provides them), we believe strongly that tools also need to directly model and expose higher-level musical concepts. Unfortunately, such tools are somewhat rare.

Our earlier work demonstrated the use of graphical shapes to modify musical aspects, such as melody, amplitude, or tempo (Abrams et al. 1999). Shapes can be applied anywhere in the compositional hierarchy and modify the music in subtle ways by specifying expressive detail, or by transforming the underlying score into what is perceived as new musical material. Our representation of functional tonality enables us to apply these tools on pitch while retaining the functional relationships; i.e. the music remains pleasing to the ear (Abrams et al. 2000).

For QSketcher, our emphasis was on the issues of creative workflow (e.g. the Idea Space, visual layouts, context management and the like). More extensive manipulation facilities will be added in a later phase. As noted above, basic event editing tools are now in place; manipulating the compositional structure is supported in the Idea Space and Music Schematic.

Composing for a visual medium involves the special challenge of managing the relationship between time and frames, and the fact that this relationship often changes during composition, as film editing progresses. Thus, the composer needs high level tools to adjust the music accordingly (again using macro and micro tools as needed).

QSketcher provides several novel mechanisms for dealing with time. First, each bundle in the Idea Space can have its own tempo map and time signature. Second, a bundle can be stretched by dragging its right-hand boundary in one of two modes. Using the left-mouse button to drag, modifies the bundles length, leaving the tempo unchanged.

Using the right-mouse button scales the bundles tempo map, thus modifying the bundles effective duration.

The timeline also supports markers bound to specific time locations. Dragging markers with the left-mouse button alters their temporal position. Conversely, right-mouse dragging modifies the tempo map: if the map had a change at that location, the corresponding tempo is modified; otherwise, a new tempo change is introduced.

One can also align markers with key-frames in the Film Strip by simply dragging the marker onto the desired key-frame. This has one of several effects. If the marker is a bundle-beginning marker, the bundle's onset is changed to reposition the bundle at the marker time. If the marker is a bundle-end marker, the bundle's tempo map is scaled (if possible) to make the bundle end at the given time. If the marker is in the middle of the bundle, the tempo map is modified (by inserting a tempo change, if necessary, or altering an existing change) to align the marker with the key-frame. Finally, key-frames can be dropped onto the time-line, thereby creating a new marker and performing a similar tempo-map modification.

7. Music Representation

Clearly, a music representation (i.e. a set of data structures) that directly models the key concepts makes implementing all of the above easier. Our music representation is essentially a best-of-breed design, incorporating those features that facilitated support of composer-requested features. ((Dannenberg 1993) provides an excellent overview of music representation issues.)

Given the requirements of capturing all forms of ideas and organizing them in a common environment, we designed our representation based on a small number of general concepts. All materials – textual notes, modifiers, phrases, motives, graphical sketches, etc. – are “bundles”, and are stored and manipulated in the same ways. This flexibility allows the database to store different kinds of content, allows pinned and embedded views of anything to be placed anywhere, and so on. In fact, database folders are also bundles. Certain bundles (MusicBundles, for example), are “specially marked packages,” i.e., derived classes that add accelerator methods for common properties (e.g. Note's GetPitch()), and provide semantically-important processing beyond raw property access (e.g. Note::GetPitch() can take modifiers and context into account in computing the pitch).

At the heart of our music representation is a “free-form” mostly-hierarchical structure of “bundles”. The system does not force track-oriented, notation-oriented, MIDI-specific or other overly constraining models on the music (although the system is capable of representing all of these possibilities). In particular, this permits the creation of skeletal compositional structures, and is a natural match for the free-form “blank-sheet-of-paper” paradigm of the Idea Space. The composer team felt this was a key component of the system's ability to support a fluid work style in the early phases of the composition process.

Every bundle and note is a property bag, i.e., an association of symbols (“interned” strings) with properties. All musical data and meta-data are represented by properties. Valid property types include strings, symbols, numbers, booleans, pitches, temporal locations/durations, instrument descriptors, expressive curves, time-sorted event lists, wall-clock timestamps, images and references. Any client can add properties to a bundle. This feature allows the composer and the various tools to add both functional data and arbitrary annotations to bundles as well as to folders in the Database Palette. The current implementation is reasonably time- and space-efficient. Figure 7 illustrates the use of properties in a small compositional structure.

Properties can be “inherited” by child bundles from parent bundles, and overridden if desired. For example, a child can inherit or override tempo from its parent. Modifiers (see below) are also inherited, and provide a powerful, high-level mechanism for altering and shaping content in child bundles from enclosing contexts.

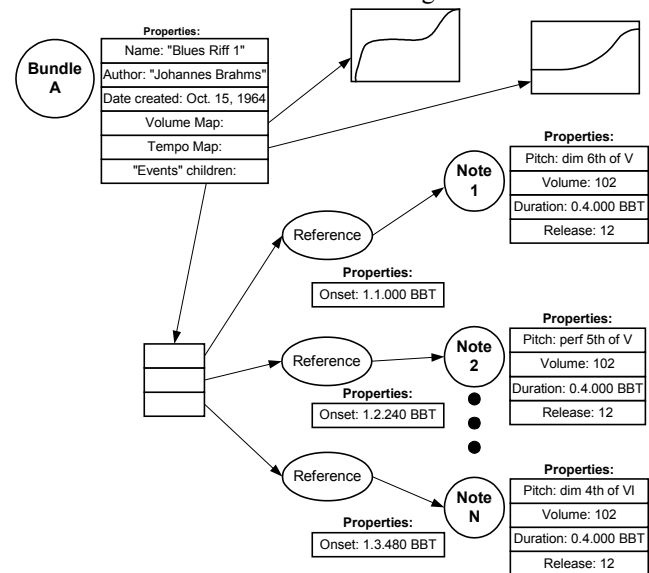


Figure 7. Property and Bundle Representation

“Colored links” are used to represent all of the various relationships among bundles. For example, parent/child relationships in event lists are represented using links of “event color”, user-defined database folders’ contents are indicated with links of “folder color”, and the ancestry of copies is recorded using “ancestry-colored” links. Any client can add links of arbitrary colors, and navigate the content using these links. The search mechanism in fact uses the same code to scan through database folders that it uses to scan through the composition hierarchy. Figure 8 illustrates the use of links in a compositional structure.

A bundle in the hierarchy can be a *reference* to a *shared* musical entity, such as a motivic riff or melody. References are themselves property bags and can therefore augment or override properties of the shared content. This was originally designed to support motivic reuse, while retaining the possibility of customizing each reuse. For instance, a

shared bundle's content could be interpreted within two distinct harmonic contexts (or tempo maps) provided by two different parents (see Figure 8). However, our composer team felt that copy-to-modify was sufficient, given the ability to trace the “ancestry” of copies. So, in keeping with our mantra “the composer is always right”, we did not expose shared references in the user interface.

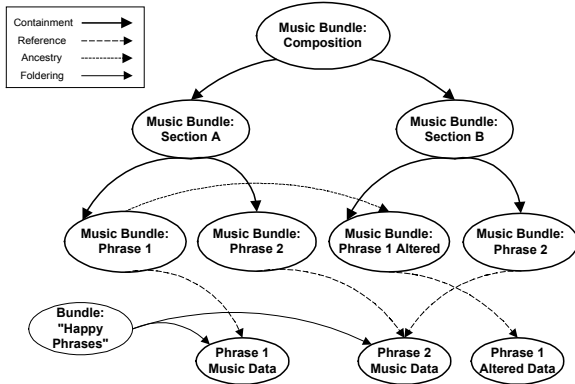


Figure 8. Bundle and Reference Representation

The music representation is easily instrumented with our pitch representation (Abrams et al. 2000), which embodies a basic model of Western tonal music. This supports “smart” transpositions, harmonic transformations, and melodic shaping, while preserving functional aspects of harmony. Expressive curves (“modifiers” (Abrams et al. 2000)) can be used to modulate any numeric parameter, such as pitch, volume, onset, duration, or tempo. These curves are a key part of the conceptual framework that our composer team described in the early compositional process.

The system supports several different types of time: bar/beat/tick, SMPTE, microseconds, and audio samples. A note's onset, for example, can be specified in any form of time that is defined within its context, i.e., within the enclosing bundles. This ability is key to describing, e.g., musical content with a metric structure (bars/beats/ticks), locked to specific SMPTE frames within the film, one of our composers' core functional requirements. As a further illustration, a bundle with a metric structure can hold a note whose onset lies on a particular frame, to align the note with a particular film event, regardless of tempo changes. Compound representations such as bar/beat/tick can house a mixture of positive and negative quantities, allowing the expression of “50 ticks before beat 2 of bar 3”.

Music representation objects issue notifications to interested clients (e.g., the user interface) for relevant events. For example, property bags (notes and proper bundles) emit notifications as property values change. Since event lists are properties, adding and deleting children uses this mechanism to keep clients updated.

8. Conclusions and future work

We have outlined a new compositional system that is novel in three ways. First, its design is strongly rooted in

musical concepts. Second, it supports the compositional workflow in several ways: by helping the composer to capture ideas, to organize those ideas to focus on the essential aspects, and to manipulate those ideas in intuitive ways. Third, the system helps the composer to keep track of her state-of-mind as he shifts from one activity to another.

The system is an work-in-progress, but the composers involved in this project are actively providing valuable feedback and helping us to refine it, and to maintain the conceptual focus where it belongs: on the composer.

In music, as in many other domains, information technology tools can have a significant impact on the creative process. Tools can reify certain artistic practices, while marginalizing others. That is, many tools encapsulate the particular practices used (or witnessed) by the tool developers, and force users to adopt that terminology, structure, and even workflow. When this happens, the tool is certainly influencing the worker - and, most likely, limiting his ability to explore the space of creative possibilities. In this research, we identified some techniques that help prevent this common pitfall, freeing the composer to explore the musical space, think and work in common musical concepts, and freely move among the various modes of work in a fluid manner.

More generally, we have found the music domain to be an excellent vehicle for developing new metaphors and mechanisms for supporting creativity, and believe that these ideas apply to many other domains. The key aspects of tools needed to support creative workflow (i.e. capture, organize, and manipulate), and the ways in which they relate to the cognitive aspects of creativity (i.e. context, relationships, state, and domain-concepts), have the potential to grow into a general model for framing research and development in tools that support creativity across many domains.

9. References

- Abrams, S, Oppenheim, D., Pazel, D., Wright, J. 1999. “Higher-level Composition Control in Music Sketcher: Modifiers and Smart Harmony”, *Proceedings of the ICMC*, Beijing, China.
- Abrams, S., Fuhrer, R., Oppenheim, D., Pazel, D., Wright, J. 2000. “A Framework for Representing and Manipulating Tonal Music.” *Proceedings of the ICMC*, Berlin, Germany.
- Dannenber, R. 1993. “Music Representation Issues, Techniques and Systems”, *Computer Music Journal*, 17(3) (Fall 1993), <<http://www.cs.cmu.edu/~rbd/subjbib2.html>>
- Oppenheim, D. 1991. “Towards a Better Software-Design for Supporting Creative Musical Activity (CMA)”, *Proceedings of the ICMC*, Montreal, Canada.
- Stravinsky, I. 1942. *Poetics of Music*, Harvard University Press. Chapter 3, “The Composition of Music.”
- Tulving, E. 1972. “Organization of Memory”, Academic Press.
- Tulving, E. 1983. “Elements of Episodic Memory”, Oxford University Press.
- Wright, M., Wessel, D., Freed, A. 1997. “New Musical Control Structures from Standard Gestural Controllers”, *Proceedings of the ICMC*, Thessaloniki, Greece.