Using the Creative Process for Sound Design Based on Generic Sound Forms

Guerino Mazzola, Florian Thalmann
School of Music, University of Minnesota
2106 Fourth Street South
Minneapolis, MN 55455

Abstract

Building on recent research in musical creativity and the composition process, this paper presents a specific practical application of our theory and software to sound design. The BigBang rubette module that brings gestural music composition methods to the Rubato Composer software was recently generalized in order to work with any kinds of musical and non-musical objects. Here, we focus on time-independent sound objects to illustrate several levels of metacreativity. On the one hand, we show a sample process of designing the sound objects themselves by defining appropriate datatypes, which can be done at runtime. On the other hand, we demonstrate how the creative process itself, recorded by the software once the composer starts working with these sound objects, can be used for both improvisation with and automation of any defined operations and transformations.

Keywords Rubato Composer, Synthesis, Sound Design, Creativity.

Introduction

Musical creativity in composition is a complex activity spanning from symbolic shaping of score symbols to the design of interesting sounds. The latter poses many problems when it comes to an intuitive understanding of the ways sound waves can be produced and combined. Particularly with methods such as frequency or ring modulation, it is difficult for a composer to see associations between structural aspects and the resulting sound (Dahlstedt 2007, p. 88). Furthermore, interfaces commonly used for sound design are typically close to the physical reality of sound production, such as the typical interface of a modular synthesizer with knobs, buttons and patch cords, and lack aspects of spacial imagination crucial for the understanding of complex structures.

A variety of approaches have been suggested that enable sound designers to overcome some of these obstacles without having to acquire a background in acoustics, signal processing, or computer programming. Systems that enable parameter randomization, for instance, can lead to discoveries by chance, but they can then not be further processed without knowledge of technical details. Evolutionary systems, alternatively, can be helpful by allowing composers to automatically generate new sounds based on selected parental sounds and thus judge on a purely aesthetic level (Dahlstedt 2007). However, the processes behind such generation are typically hidden and the outcome difficult to understand. The same is true for artificial intelligence and learning system approaches that have also been proposed, such as in (Miranda 1995).

In this paper we illustrate how the creative process of sound design itself can be used for sound design. We start out by briefly summarizing our recent research on and model of the creative process. Then, after a short introduction to the principles of the BigBang rubette module, we discuss the characteristic problems of sound design and show how solutions can be found with our software. These solutions illustrate how metacreativity can take place on several levels of both synchronic and diachronic nature. This will lead us to a solution which unites all common synthesis methods and can still be used in the same intuitive way. Finally, we explain how our current version of BigBang enables composers to retrospectively use their own creative design process to reshape the result they arrived at and find neighboring sounds in the sense of Boulezian analyse créatrice.

The Creativity Process Scheme

The theoretical model of the creative process on which the discussions in this paper are based consists in a semiotic approach. It was first described in (Mazzola, Park, and Thalmann 2011) and later applied in (Mazzola and Park 2012; Andreatta et al. 2013) and consists in a sequence of seven steps that can be summarized as follows:

1. Exhibiting the *open question*
2. Identifying the *semiotic context*
3. Finding the question's *critical sign or concept* in the semiotic context
4. Identifying the concept's *walls*
5. *Opening* the walls
6. Displaying *extended wall perspectives*
7. *Evaluating* the extended walls

In this model, creativity implies finding a solution to the open question stated in the initial step, and which must be proven to be viable in the last step. The contextual condition guarantees that creativity is not performed in an empty space. It is not a formal procedure as suggested by other scholars, such as David Cope (Cope 2005), but generates new signs with respect to a given meaningful universe of signs. The critical action here is the identification of the critical sign's "walls", its boundaries which define the famous "box", which creativity would open and extend.

The model has been applied to many general examples (Mazzola, Park, and Thalmann 2011), such as Einstein's annus mirabilis 1905 when he created the special relativity theory, or Spencer Silvers discovery of 3Ms ingenuous Post-It in 1968. Relating more specifically to musical creativity in composition, the authors discussed the creative architecture of Ludwig van Beethoven's six variations in the third movement of op. 109 in the light of our model, our analysis being confirmed by Jürgen Uhde's (Uhde 1974) and William Kinderman's (Kinderman 2003) prestigious analyses of op. 109, as well as the creative analysis of Pierre Boulez's structures pour piano I, a computer-aided creative reconstruction of that composition in the sense of Jean-Jacques Nattiez's paradigmatic theme (Nattiez 1975) and of what Boulez calls analyse créatrice in (Boulez 1989), starting from György Ligeti's famous analysis (Ligeti 1958).

Walls in Sound Design

The creative process plays several roles in the context of this paper. Most importantly, the software presented later builds on recording each step in a more abstract way and giving the composers access to refine their decisions at a later stage. However, it can be found on other (meta-)compositional levels such as the process of defining sound object types before composing, and even on the level of finding a new solution for sound design. The latter will be discussed here as an example.

In the light of this model, the common problems of sound design stated in the introduction can be identified to consist in *multiple "walls"*. First, formulas describing sounds are anything but intuitive and can typically not be handled by the mathematically non-trained composer. Second, interfaces are usually modeled on the physical process of sound synthesis and may thus seem unintuitive and laborious to manipulate. Third, sound manipulation usually consists in manually altering single parameters sequentially, or simultaneously altering several parameters in a discrete and uncontrollable automated way. Fourth, there are very different ways of creating formulas and generally no unified presentation of Fourier, FM, wavelet synthesis is given. This double wall is conceptual and representational: the corresponding (double) *open question* could be formulated as follows: *Can we design a generic formula for basic methods of sound design and represent its instances in a way that allows for intuitive, continuous, and accessible interaction?* The following sections lead to solutions that can be achieved using the *denotator* formalism implemented in the music software Rubato Composer. To prove their viability they will be *evaluated* in practice with examples generated in BigBang.

Denotators, Rubato Composer, and BigBang

Rubato Composer, a Java software environment and framework (Milmeister 2009), is based on recent achievements in mathematical music theory, which includes the versatile formalism of *forms and denotators* that roughly corresponds to the formalism of classes and objects in object-oriented programming but is realized in a purely mathematical way based on topos theory. *Forms* are generalized mathematical spaces commonly based on the category of modules $Mod^{@}$ and created using several logical-structural types including **Limit** (product), **Colimit** (coproduct), **Power** (powerset). The basic spaces, corresponding to primitive datatypes, are referred to as **Simple**. *Denotators*, instances of these forms, can be seen as points in the corresponding spaces. They are the basic data type used for the representation of musical as well as non-musical objects in Rubato Composer. Rubette modules in the software typically operate on them by applying transformations, so-called *morphisms*, or evaluating them using *address changes*. For details, refer to (Mazzola 2002; Milmeister 2009).

The BigBang rubette module (Thalmann and Mazzola 2008; 2010; Mazzola and Thalmann 2011) applies insights from transformational theory (Lewin 19872007; Mazzola and Andreatta 2006), music informatics, and cognitive embodiment science (Mazzola, Lubet, and Novellis 2012) by implementing a system of communication between the three musico-ontological levels of embodiment (facts, processes, and gestures) (Thalmann and Mazzola 2011). Traditionally, a composition is seen as a definite *fact*, a static result of the composition process. In BigBang it is reinterpreted as a dynamic *process* consisting of an initial stage followed by a series of operations and transformations. This process, in turn, is enabled to be created and visualized on a *gestural* level. The composition can thus typically be represented on any of the three levels. As a number of multi-dimensional points (denotators) in a coordinate system (according to the form) on the factual level, a directed graph of operations and transformations on the processual level, and a dynamically moving and evolving system on a gestural level. BigBang implements standardized translation procedures that mediate between these representations and arbitrarily translate gestural into processual compositions, processual into factual ones, and vice versa.

More precisely, BigBang enables composers to draw, manipulate, and transform arbitrary objects represented in denotator form in an intuitive and gestural way and thereby automatically keeps track of the underlying creative process. It implements a powerful visualization strategy that consists in a generalization of the piano roll view, which can be recombined arbitrarily and which works for any arbitrary data type, as will be explained later (Figure 1). In the course of composition, any step of generation, operation, and transformation performed on a gestural input level is recorded on a processual level and visualized in form of a transformational diagram, a directed graph, representing the entire composition (shown in Figure 2). Furthermore, composers cannot only interact with their music on an immediate gestural level, but also oversee their own compositional process on a more abstract level, and even interact with this process

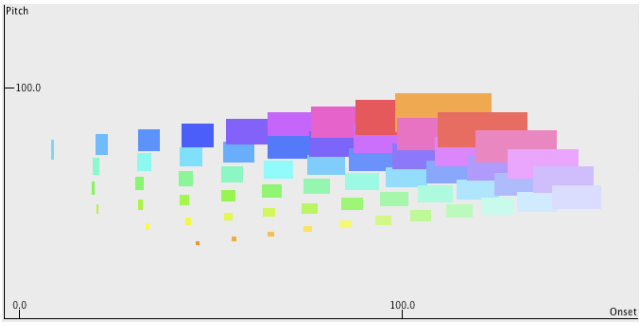

Figure 1: A factual representation of a composition in BigBang. Each of the rectangles represents a specific object, having a number of freely assignable visual characteristics such as size, position, or color.

by manipulating the diagram in the spirit of Boulezian analyse créatrice (Boulez 1989). If they decide to revise earlier compositional decisions, those can directly be altered, removed from the process, or even inserted at another logical location.

Finally, BigBang can visualize the entire compositional process in animation, i.e. generate a "movie" of the transformational evolution of a composition. This tool is of great benefit for the metacreative control of the compositional process in music. The composer can trace back to the moments in the creative process where the walls of critical concept boxes were opened, and take revise decisions taken after that.

Generic Sound Forms

Before showing how sample datatypes (forms) for sound design can be created in practice, it will be helpful to introduce the solution to one of the discussed "walls" to be used as a reference, the one of different synthesis methods or formulas. This section presents a unified format that allows for all common basic synthesis methods to be combined. First, in addition to previously discussed aspects and techniques concerning forms and denotators, we introduce just one very useful definition technique of partial evaluation. It works as follows. Suppose that a form F is defined which in its coordinator forms refers to an already given form G. Then we may replace G by a denotator $D : A@G$ at address A, for all denotators we want to build of form F. We may therefore define a partially evaluated form $F(D)$ which looks exactly like F, except that instead of G, we insert D. This is a well-defined procedure even without having previously defined F since D has a unique reference to its form G. This technique is very useful to create general forms that specialize to more specific forms, as we shall see in the following section.

Here we focus on conceptual design of sound forms, i.e. forms which capture concepts that are necessary to create sound architectures. The challenge of such an endeavor is to navigate between too general approaches which include special cases, but without any specific tools to exhibit special cases, and too special approaches that eventually appear as items in a disconnected list. The too general approach

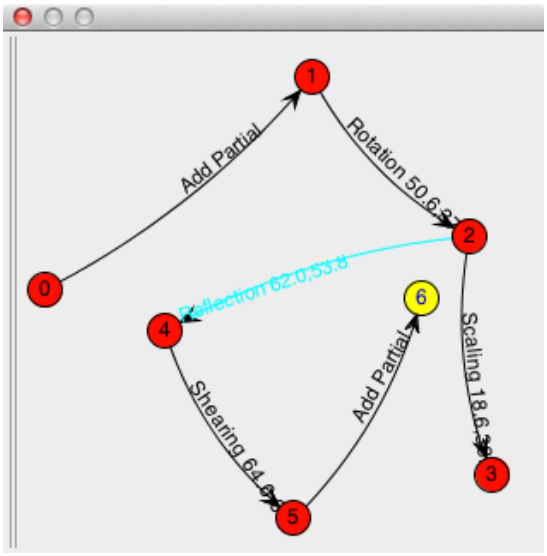

Figure 2: A graph of a composition process of a $SoundSpectrum$ including all five geometric transformations (*Translation, Rotation, Scaling, Shearing, Reflection*) as well as the drawing operation (*Add Partial*).

would be to describe any sound by its physical appearance, namely as a function $f(t)$ of time t that is given in an adequate discretization and quantization, but without and further definition of how such functional values may be generated. The opposed, too special, approach could for example consist of a Fourier synthesis form, an FM synthesis form and a wavelet synthesis form, building a three-item list without internal connections or visible generating principle.

The point of a generic sound form design is the same as for concept design with forms and denotators in general: The design must be open towards new forms, but their building rules must be precise and specific enough to guarantee efficient building schemes. Moreover, the sound forms must also have a basis of "given" sound forms, much like the mathematical basis (category of modules) in general form construction as implemented in Rubato Composer. We shall however permit an extensible sound form basis, meaning that, like it is generally admitted for forms, each new sound form will be registered among the set of given sound forms. Following the general rule for forms, we shall also require unique names, no homonyms are permitted. This means that we can define a SoundList form as follows (the notation follows the denotex standard (Mazzola 2002, p. 1143)):

$$SoundList : .\mathbf{List}(SoundName),$$
$$SoundName : .\mathbf{Simpl}(\langle UNICODE \rangle).$$

All sound forms named in this container will be accessible for sound production. To be precise, sound forms have to produce "time" functions $f : \mathbb{R} \to \mathbb{R}$. Their physical realization is the job of sound generator soft- and hardware and must be implemented by a general program that takes care of discretization and quantization.

The generic sound form we propose here is defined by

$GenericSound$: .**Limit**$(NodeList, Operation)$,
$NodeList$: .**List**$(Node)$,
$Node$: .**Limit**$(AnchorSound, GenericSound)$,
$AnchorSound$: .**Limit**$(SoundList, Position)$,
$Position$: .**Simpl**$(\mathbb{Z})$,
$Operation$: .**Simpl**$(\mathbb{Z}_3)$.

These forms have the following meaning: The $GenericSound$ form presents lists of sounds, given in $Node$ form. They are combined according to one of three operations on the respective sound functions in the list, consisting of either their addition, their point-wise multiplication, or their functional composition. These three options are parametrized by the three values of $\mathbb{Z}_3$. Each sound node specifies its anchor sound, which is a reference to an item of the given sound list, and it also specifies "satellite" sounds, which are given as a denotator of form $GenericSound$.

So the $GenericSound$ form is circular, but its denotators are essentially lists of lists of lists... that eventually become empty, so no infinite recursion will happen for practical examples. Another more critical circularity can occur if a sound definition refers to the sound being defined in the list of given sounds. This happens quite often in FM synthesis and is solved by a well-known reference of sound evaluations at earlier time units.

Basic Examples

To get a general sinusoidal function, we suppose that the sound list contains the function $\sin(t)$, named Sin, and constant functions A. The arguments of this function are defined by the function form

$$Arg : \textbf{.Limit}(Frequency, Index, Phase)$$

where all these coordinators are simple forms with real values. Denotators $arg : @(f, n, Ph)$ of this form define functions $2\pi nft + Ph$. Then the function $A\sin(2\pi nft + Ph)$ results from the functional composition of the the argument function arg with the product of constant A with Sin. Adding a list of such sinusoidal functions yields Fourier synthesis sound forms. If one adds a modulator function defined by a satellite at a node to the arg function, FM synthesis results.

The partial evaluation technique described above is an excellent tool for generating sound denotators that have specific parameters in their arguments, such as sinusoidal functions as anchor arguments.

Ring modulation evidently results from multiplying given sound functions. If one supposes that envelope functions are in the sound list, one may use the product operation to create wavelets. The sum of baby wavelets defined by discrete transform of a father wavelet enables wavelet synthesis.

If a sample sound function is added to the sound list, one may also include it in the generation of derived sound functions. Since the generic form does not restrict sound construction to classical sinusoidal waves, one may also define Fourier or FM synthesis using more general generating functions than Sin, in fact any function taken from the sound list will do.

Sound Design in Practice with BigBang

The definitions above provide a format suitable for the definition of any arbitrary sound based on additive, multiplicative and modulative synthesis. In practice, the definition of such constructs can be tedious and a visual and dynamic method can be enormously helpful, especially when results is expected to be manipulated in real-time with constant sound feedback. This section shows how this "wall", the one of continuous and consistent visual representation, was opened with the BigBang rubette.

Visual Representation and Transformation

A few remarks on the general visual and interactive concept of BigBang are necessary here, but for a more thorough discussion see (Thalmann and Mazzola forthcoming). A recent generalization of the BigBang's previous visualization concept enables visual representation of any denotator of an arbitrary form. Already in earlier versions (Thalmann and Mazzola 2008), the basis of the concept is the association of a number of *view parameters* (e.g. X-Position, Y-Position, Width, Height, Opacity, or Color) with the **Simple** denotators present in the given denotator. For instance, to obtain a classical piano roll representation, we typically associate X-Position with time ($Onset$), Y-Position with $Pitch$, Width with $Duration$, Opacity with $Loudness$, and Color with $Voice$. Any possible pairing is allowed, which can be useful to inspect and manipulate a composition from a different perspective. Furthermore, several of these perspectives can be opened at the same time, and while being transformed the composition can be observed from different perspectives simultaneously.

For the representation of denotators of arbitrary compound forms, we need to make a few more general definitions:

1. The general visualization space consists of the cartesian product of all **Simple** forms appearing anywhere in the anatomy of the given form. For instance, for a $MacroScore$ denotator of any hierarchical depth, this is $Onset \times Pitch \times Loudness \times Duration \times Voice$.
2. Any **Simple** form X the module of which has dimension $n > 1$ is broken up into several modules $X_1, \ldots, X_n$. The visual axes are named after the dimension they represent, i.e. X_n, or X if $n = 1$.
3. **Power** denotators anywhere in the anatomy define an instantiation of distinct visual objects potentially represented by view parameters. Objects at a deeper level, i.e. contained in a subordinate powerset, are considered *satellites* of the higher-level object and their relationship is visually represented by a connecting line. For example, in a $SoundScore$ (Thalmann and Mazzola 2010) we previously distinguished satellites and modulators. Now both are considered satellites, however at different logical positions of the denotator, and they are no more distinguished in a visual way.
4. Given a view configuration, the only displayed objects are denotators that contain *at least one* **Simple** form currently associated with one of the visual axes.

As in previous versions of BigBang, transformations can be applied to any selections of visible screen objects, be they of the same type or not. For instance, in a composition based on a $GeneralScore$ containing denotators of both forms $Note$: .**Limit**$(Onset, Pitch, Loudness, Duration, Voice)$ and $Rest$: .**Limit**$(Onset, Duration, Voice)$, from the perspective of $Onset \times Duration$, both $Notes$ and $Rests$ can be transformed simultaneously.

A Few Simple Sound Synthesis Examples

To illustrate the potential of the new version of BigBang a few simple examples will be helpful. Even though the generic form for sound objects described earlier elegantly unites several common methods of sound synthesis, for certain practical purposes, in order to work faster and in a more reduced way, simpler forms may be more suitable. Since in Rubato Composer new forms can be defined at runtime, users can spontaneously define data types designed for any specific purpose. They can then immediately start working with this form in the BigBang rubette, and define denotators and manipulate them as quickly and intuitively as it was possible with $Scores$ in the previous version.

A requirement for working flexibly with different formats is to ensure that they share as many **Simple** forms as possible in order to be represented in relation to each other. For instance, in view of the to date most commonly used form $Score$, it seems reasonable to define sound forms based on $Pitch$ and $Loudness$ as well, rather than frequency and amplitude. As will be exemplified later, this has the advantage that such forms can be represented and manipulated in the same coordinate system as $Scores$, for the reasons described in the previous section.

This is the second level on which a creative process can be observed, this time controlled by the sound designer. The most straightforward way is to start with a basic format that is as simple as possible, for instance the simple form $Pitch$. It seems exaggerated to speak of sound design when just working with a single **Simple** form. All we can do is define and transform one pitch at a time, a typical "wall". The most obvious way to open it is probably to extend the form in order to work with sets of pitches or clusters of sound. Formally, all we need to do is put the form in a **Power**, which leads us to $PitchSet$: .**Power**$(Pitch)$.

We just obtained our first sound object, but again we face a "wall" by not being able to control the amplitude of the object's parts. Inserting a **Limit** does the trick and leads to the following structure:

$$SoundSpectrum : .\textbf{Power}(Partial),$$
$$Partial : .\textbf{Limit}(Loudness, Pitch).$$

We obtain a constantly sounding cluster based on only two dimensions, as shown in Figure 3.

This form, however, is not well suited for the creation of harmonic spectrum, as we would have to meticulously arrange each individual pitch so that it sits at a multiple of a base frequency. To break this "wall", the following form will

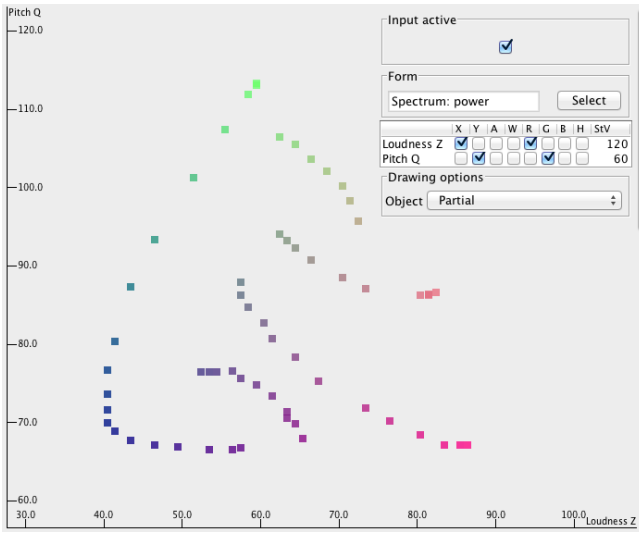

Figure 3: A sample $SoundSpectrum$.

be useful:

$$HarmonicSpectrum : .\textbf{Limit}(Pitch, Overtones),$$
$$Overtones : .\textbf{Power}(Overtone),$$
$$Overtone : .\textbf{Limit}(OvertoneIndex, Loudness),$$
$$OvertoneIndex : .\textbf{Simple}(\mathbb{Z}).$$

Figure 4 shows an example containing several such spectra, which was done by simply defining a form $HarmonicSpectra$: .**Power**$(HarmonicSpectrum)$. Since satellites $(Overtone)$ and anchors $(HarmonicSpectrum)$ do not share **Simple** dimensions, they can only be visualized if one **Simple** of each is selected as axis parameters, here $Pitch \times OvertoneIndex$. However, they can both be transformed in arbitrary ways on such a plane. This is the simplest way of working with additive synthesis in BigBang. All oscillators are supposed to be based on the same wave form and a phase parameter is left out for simplicity. This is equally the case for the following examples.

Even though the previous form leads to more structured and visually appealing results, we limited ourselves to purely harmonic sounds, since all $Overtones$ are assumed to be based on the same base frequency $Pitch$. To make it more interesting, we can decide to unite the sound possibilities of $SoundSpectrum$ with the visual and structural advantages of $HarmonicSpectrum$ by giving each $Overtones$ its own $Pitch$ by defining a form such as:

$$DetunableSpectrum : .\textbf{Limit}(Pitch, Overtones),$$
$$Overtones : .\textbf{Power}(Overtone),$$
$$Overtone : .\textbf{Limit}(Pitch, OvertoneIndex, Loudness).$$

Since values reoccurring in satellites are typically defined in a relative way to the corresponding ones of their anchor, we get the opportunity to define changes in frequency rather than the frequency themselves. A displacement of a satellite on the $Pitch$ axis with respect to its anchor enables us to

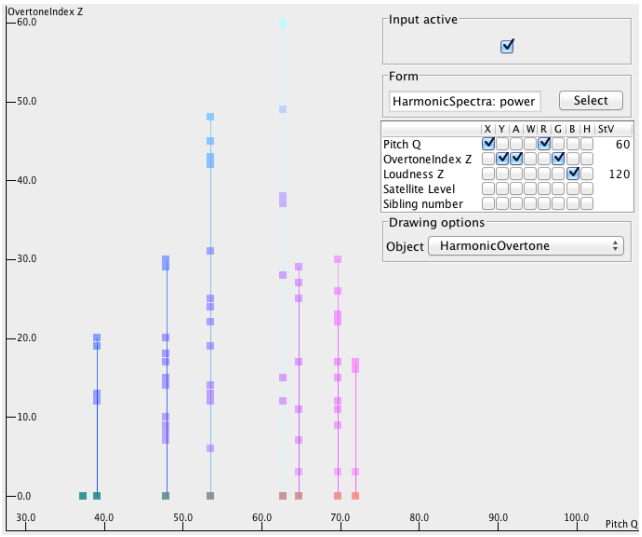

Figure 4: A constellation of eight $HarmonicSpectra$ with different $Pitches$ and $Overtones$.

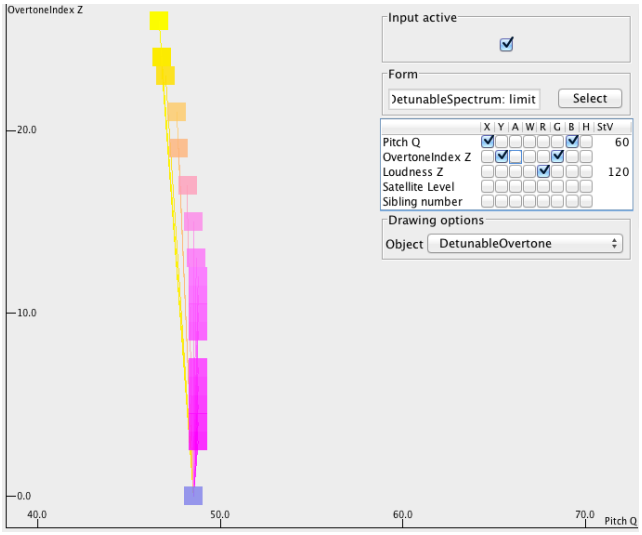

Figure 5: An instance of a $DetunableSpectrum$, where the fundamentals of the $Overtones$ are slightly detuned.

detune them. Figure 5 shows an instance of such a spectrum.

The three forms above are but three examples of an infinite number of possible definitions. Already slight variants of the above forms can lead to significant differences in the way sounds can be designed. For instance, generating complex sounds with the above forms can be tedious as there are many possibilities to control the individual structural parts. A well-known method to achieve more complex sounds with much fewer elements (oscillators) is frequency modulation, which can be defined as follows in a recursive way:

$$FMSet : .\textbf{Power}(FMNode),$$
$$FMNode : .\textbf{Limit}(Partial, FMSet),$$

where partial is defined above. Examples as complex as the one shown in Figure 6 can be created this way. Frequency modulation, typically considered highly unintuitive in terms of the relationship of structure and sound (Chowning 1973), can be better understood with a visual representation such as this. All carriers and modulators are show respective to their frequency and amplitude and can be transformed simultaneously and in parallel, which has great advantages for sound design compared to old-fashioned synthesizers and applications.

An Example of the Generic Sound Form

At this point it is worthwhile reassessing the power of our generic sound form defined earlier. The possibilities of form construction allow us to define sets (**Power**), products (**Limit**), or coproducts (**Colimit**) of any two forms defined, which allows us to use them concurrently. For instance, a **Limit** of $SoundSpectrum$ and $Score$ allows us to create compositions containing both constantly sounding pitches and notes with a certain $Onset$ and $Duration$. Figure 7 shows an example of such a composition. This way, any number of synthesis methods and musical formats can be joined. However, in order to have the possibility to logically combine synthesis methods, a cyclical form such as $GenericSound$ is essential. It allows us to generate structures where for instance sound objects are generated via ring modulation, combined in an additive way, and finally used to modulate a carrier object.

Now how can our $GenericSound$ form be used in practice? Besides allowing for sounds to be built using combinations of all of the above synthesis methods, this format also draws on a repertory of any kind of sound waves given in $SoundList$. Nevertheless, a crucial requirement for maximal flexibility and intuitiveness in practice is to use basic sounds that are based on a common space. The most straight-foward example is to use standard functions such as sinusoidal, triangular, square, or sawtooth waves similar to the definition of a sine function given earlier. They all share a 4-dimensional space spanned by amplitude A, frequency f, index n, and phase Ph. For simplicity, again, we may ignore Ph and obtain $AnchorSounds$ represented in a three-dimensional space $Loudness \times Pitch \times OvertoneIndex$. $GenericSounds$ in turn are represented by their $Operation$. A concrete $Oscillator$ form corresponding to this and replacing the $AnchorSound$ above could be defined as follows:

$$Oscillator : .\textbf{Limit}(Loudness, Pitch, OvertoneIndex, WaveForm),$$
$$WaveForm : .\textbf{Power}(\mathbb{Z}_4),$$

where $WaveForm$ points to any of the four basic shapes above (sinusoidal, triangular, square, sawtooth). Figure 8 shows an example of this form in use in BigBang.

Using the Creative Process for Sound Design

Now we arrive at the central point of our paper. Even though the process of inventing the sound objects to work with was

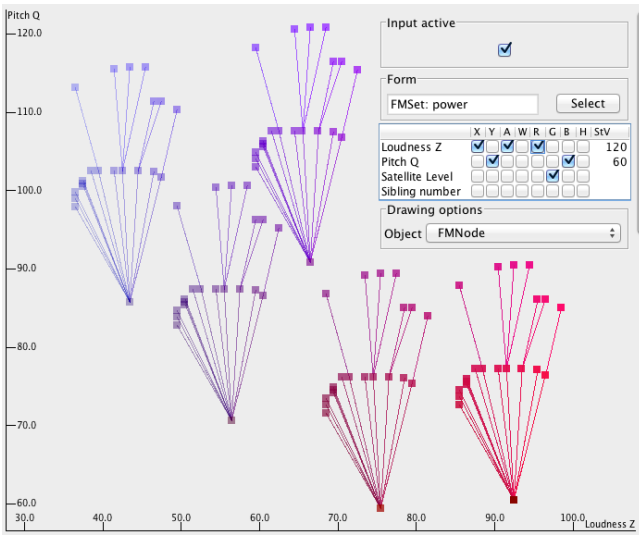

Figure 6: An $FMSet$ containing five carriers all having the same modulator arrangement, but transposed in $Pitch$ and $Loudness$.

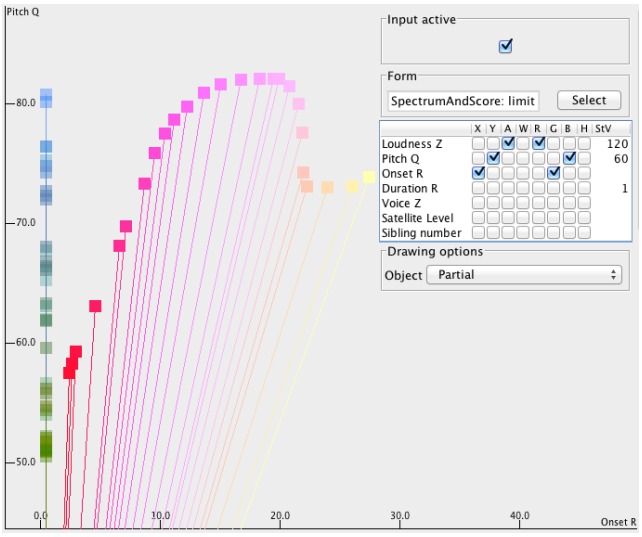

Figure 7: A composition based on a **Limit** of a $SoundSpectrum$ ($Pitches$ at $Onset$ 0) and a $Score$ ($Pitches$ on the right hand side).

already connected to our creative process, there is a much more direct way in which such a process itself can be used to design sound. As mentioned earlier, BigBang remembers anything composers do as soon as they start working with defined forms. For example, if we decide to work with a $SoundSpectrum$, we can draw visual objects on a two-dimensional plane, each of them representing a sound object. Then, we can select subsets of these objects and scale, rotate, shear, reflect, or translate their position, generate regular structures (wallpapers) with them, or use forcefield-like methods to alter them (Thalmann and Mazzola 2008). The resulting construct is a graph as shown in Figure 2. By selecting a node, composers can go back to any previous compositional state and by selecting an edge, they can gesturally alter previous transformations while observing the effect on whichever state is selected.

Used in a conscious way, this functionality enables sound designers to predefine a sequence of transformations they are interested in and only later refine them, by continuously adjusting the end result. They can thus first improvise by experimenting broadly with any of the available transformations until they reach a preferred sound, and later equally dynamically travel through all sounds in the neighborhood of the their result. Thereby they are not changing single parameters in a linear way as with common synthesizer interfaces, but changing multiple parameters in a complex way, such as for instance rotating both frequency and amplitude of hundreds of oscillators around a defined sound center.

MIDI-Controllers and Gestural Animation

There are several possibilities of using the constructs and procedures defined in the previous and this section in practice. To work with $SoundSpectrum$ in an independent manner, for instance, we can simply let BigBang's synthesizer play the sounds continuously and then add or remove remove partials and transform them in an improvised fashion.

Some of the more pitch-oriented forms can, however, easily be used to be triggered by a MIDI-controller. $HarmonicSpectrum$ for example has a clearly defined base frequency, which can be used to transpose the sounds relatively to fit the keys of a keyboard and be used in the fashion of a traditional keyboard synthesizer. Even non-harmonic sounds such as the ones defined by a $SoundSpectrum$ or an $FMSet$ could be used this way, by mapping the designed sound to the key corresponding to its median point, lowest and loudest note, or the like, and transposing it for all other keys.

Even during the process of sound design, a controller can be helpful. For instance, BigBang enables all transformations to be modified using control change knobs by mapping them in order of application. This way, a practically oriented sound designer can focus on playing and listening rather than switching back and forth between instrument and computer. This can of course be used in performance as well.

To exploit the full potential of BigBang for sound design, one crucial aspect needs yet to be considered. The graph generated during the process of sound design can also be brought back in a gestural form by recreating each transformation in time, using Bruhat Decomposition (Mazzola and Thalmann 2011). What results is a sonified animation of the sound's evolution. Obviously, if our composition does not contain any temporal element as such, as it is for all the forms introduced above, it can be useful to bring in a temporal aspect this way. By editing the evolutionary diagram, the composer can thus not only design the sound as such, but obtains a temporal dimension, which can mean that the movie of the evolutive process becomes the actual composition. Again, sounds designed this way could be triggered by MIDI-controllers as described above, which leads to richer

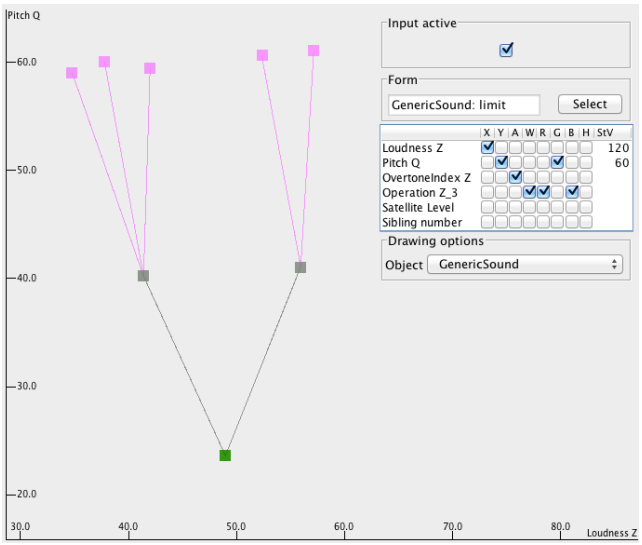

Figure 8: A generic sound using all three methods of synthesis.

and more lively sound capabilities.

Conclusion

This paper took the sound design capabilities of the BigBang rubette to illustrate the application of our theory of the creative process on several compositional levels. First, a practically viable solution has been presented that open the multidimensional walls of sound design presented in the beginning of the paper. Unified format, instructive spacial visualization, and continuous, intuitive, and multidimensional manipulation present an extended space of experimentation. Second, the process was used to describe sample preparatory thoughts when it comes to defining the data types suitable for various situations, all either derived from or specialized versions of the unified format. Third, the methods were described with which the creative process during the act of sound design can be used to modify or refined the resulting sound in a meaningful way. The benefits are comparable to other approaches such as evolutionary or artificial intelligence systems, however, with the difference that designers have possibilities to deliberately influence the outcome that go beyond purely aesthetic decisions, without having to understand the underlying formulas in detail.

References

Andreatta, M.; Ehresmann, A.; Guitart, R.; and Mazzola, G. 2013. Towards a categorical theory of creativity for music, discourse, and cognition. In *Proceedings of the MCM13 Conference*. Heidelberg: Springer.

Boulez, P. 1989. *Jalons*. Paris: Bourgeois.

Chowning, J. 1973. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society* 21.

Cope, D. 2005. *Computer Models of Musical Creativity*. Cambridge, MA: MIT Press.

Dahlstedt, P. 2007. Evolution in creative sound design. In Miranda, E. R., and Biles, J. A., eds., *Evolutionary Computer Music*. Springer. 79–99.

Kinderman, W. 2003. *Artaria 195*. Urbana and Chicago: University of Illinois Press.

Lewin, D. 1987/2007. *Generalized Musical Intervals and Transformations*. New York, NY: Oxford University Press.

Ligeti, G. 1958. Pierre Boulez: Entscheidung und Automatik in der Structure Ia. *Die Reihe* 4:21–32.

Mazzola, G., and Andreatta, M. 2006. From a categorical point of view: K-nets as limit denotators. *Perspectives of New Music* 44(2).

Mazzola, G., and Park, J. 2012. La créativité de Beethoven dans la dernière variation de l'op. 109 une nouvelle approche analytique utilisant le lemme de Yoneda. In Andreatta, M., et al., eds., *Musique/Sciences*. Paris: Ircam-Delatour.

Mazzola, G., and Thalmann, F. 2011. Musical composition and gestural diagrams. In Agon, C., et al., eds., *Mathematics and Computation in Music - MCM 2011*. Heidelberg: Springer.

Mazzola, G.; Lubet, A.; and Novellis, R. D. 2012. Towards a science of embodiment. *Cognitive Critique* 5:59–86.

Mazzola, G.; Park, J.; and Thalmann, F. 2011. *Musical Creativity – Strategies and Tools in Composition and Improvisation*. Heidelberg et al.: Springer Series Computational Music Science.

Mazzola, G. 2002. *The Topos of Music. Geometric Logic of Concept, Theory, and Performance*. Basel: Birkhäuser.

Milmeister, G. 2009. *The Rubato Composer Music Software: Component-Based Implementation of a Functorial Concept Architecture*. Berlin/Heidelberg: Springer.

Miranda, E. R. 1995. An artificial intelligence approach to sound design. *Computer Music Journal* 19.2:59–75.

Nattiez, J.-J. 1975. *Fondements d'une sémiologie de la musique*. Paris: Edition 10/18.

Thalmann, F., and Mazzola, G. 2008. The bigbang rubette: Gestural music composition with rubato composer. In *Proceedings of the International Computer Music Conference*. Belfast: International Computer Music Association.

Thalmann, F., and Mazzola, G. 2010. Gestural shaping and transformation in a universal space of structure and sound. In *Proceedings of the International Computer Music Conference*. New York City: International Computer Music Association.

Thalmann, F., and Mazzola, G. 2011. Poietical music scores: Facts, processes, and gestures. In *Proceedings of the Second International Symposium on Music and Sonic Art*. Baden-Baden: MuSA.

Thalmann, F., and Mazzola, G. forthcoming. Visualization and transformation in a general musical and music-theoretical spaces. In *Proceedings of the Music Encoding Conference 2013*. Mainz: MEI.

Uhde, J. 1974. *Beethovens Klaviermusik II*. Stuttgart: Reclam.