71

Facts: Denotators and Their Visualization and Sonification

The facts, or objects, that the rubette *BigBang* in RUBATO® Composer deals with are denotators, which can be considered points in the spaces defined by their forms, as introduced in Chapter 6. So far, we have only seen a small portion of the variety of forms that can be defined in RUBATO® *Composer*. However, any conceivable musical or non-musical object can be expressed with forms and denotators, many of them just with the category of modules $\mathbf{Mod}^@$. The most recent version of *BigBang* was made compatible with as many forms as possible, even ones that the users may spontaneously choose to define at runtime. In order to handle this as smoothly as possible, we had to find a suitable way of representing denotators within the rubette, which we call `BigBangObjects`.[1] In this chapter, we describe how this works.

71.1 Some Earlier Visualizations of Denotators

In order to understand the evolution of *BigBang*'s visualization system it will be helpful to look at some earlier attempts at visualizing denotators. Several dissertations were based on an implementation of denotators and forms. Stefan Göller's had visualization as its main focus and Gérard Milmeister's included a number of smaller visualization tools.

71.1.1 Göller's PrimaVista Browser

The goal of Göller's dissertation was to visualize denotators "in an active manner: visualization as navigation" [372, p.55]. The result was the sophisticated *PrimaVista Browser*, implemented in Java3D, that featured a three-dimensional visualization in which users could browse denotators in first-person perspective. *PrimaVista* could be customized in many ways using a virtual device, the *Di*, shown in Figure 71.1 [372, p.107].

PrimaVista was capable of representing any type of zero-addressed $Mod^@$ denotator as a point or a set of points in $\mathbb{R}^3$ while preserving both order and distance of the original data structure as well as possible. **Limit** and **Colimit** denotators of any dimensionality and their nested subdenotators were folded in a two-step process, first into $\mathbb{R}^n$ then into $\mathbb{R}^3$. Thereby, for any denotator d the mapping $Fold : F(d) \to \mathbb{R}^3$ had to be injective. The first step of this process mapped the values of the **Simple** denotators found in the given denotator hierarchy, regardless of their domain, into $\mathbb{R}^n$ by injecting or projecting each of the individual values into $\mathbb{R}$. A matrix defined which denotator dimensions were mapped into which of the n dimensions of the real codomain space, allowing for both multiple mappings and merging mappings. A so-called *greeking* procedure made sure that only denotator values up to a certain level of hierarchical depth were taken into account, which enabled dealing with circular structures. The second step of the process consisted in folding

[1] Every object that literally exists as a Java object in *BigBang*'s code will be written in verbatim font here, even if we just define them conceptually here.

G. Mazzola et al., *The Topos of Music III: Gestures*, Computational Music Science,
https://doi.org/10.1007/978-3-319-64481-3_16

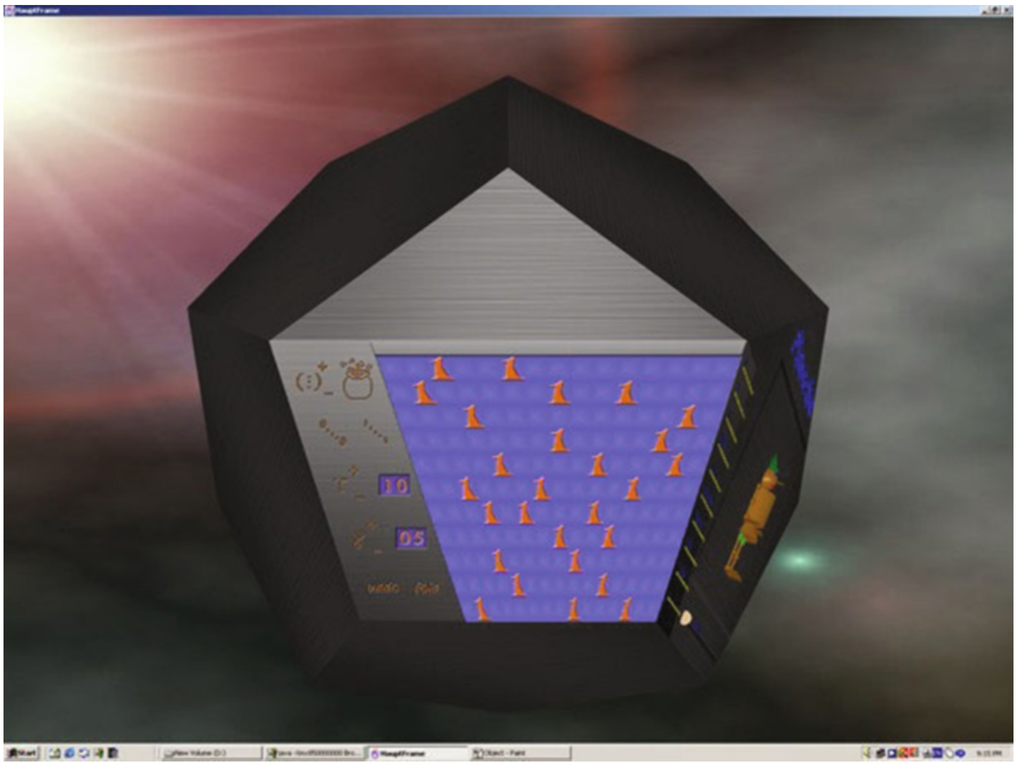


Fig. 71.1. The *Di* of Göller's *PrimaVista* browser.

the obtained $\mathbb{R}^n$ vectors into $\mathbb{R}^3$ by privileging specified dimensions and folding the remaining ones to the mantissa, the decimal digits after the comma.

Göller discussed adventurous ways of visualization, replacing the points in $\mathbb{R}^3$ with complex three-dimensional objects the parts of which he called *satellites*, not to be confused with satellites as they are defined in this context,[2] each of them representing additional characteristics of the represented denotators. Each of Göller's satellites is characterized by the following variable visual parameters: position (x, y, z), rotation vector (rx, ry, rz, α), scale (sx, sy, sz), color $(red, green, blue)$, texture, sound $(pitch, loudness, instrument, sysex)$ [372, p.77]. The most complex object finally implemented is the Pinocchio satellite shown in Figure 71.2. Göller even suggests some satellites may be moving in time to represent parameters such as frequency. This feature was, however, finally not implemented. Another feature not implemented was a generalization of the musical score, where each satellite is associated with sounds that would be played when intersected with a plane, or more generally an algebraic variety, moving in time [372, p.84-5]. Finally, Göller discusses the concept of so-called *cockpits*, where an object's subsatellites become actuators in the form of levers, buttons, or knobs, through which users can change the underlying denotator [372, p.95]. Again, this was not implemented within the scope of his thesis. In addition to this, Göller envisioned ways of transforming and manipulating objects that are similar to the ones of the *BigBang* rubette [372, p.123f].

There are several issues with Göller's approach, some of which explain the difficulties that arose when trying to implement the ideas. First, the folded spaces pose problems of ambiguity in visualization and especially transformation. If one dimension of $\mathbb{R}^3$ represents several denotator dimensions at the same time and the user starts transforming the denotator, it is not intuitively deducible from the visible movement how the denotator values are affected. Representation is often ambiguous, where differences in dimensions folded to the mantissa become only subtly visible and often visually indistinguishable from a simple projection.

[2] See Sections 71.2 and 71.3, where satellites are defined as elements of sub-powersets of a denotator. Also, in Göller's work, there are only two levels: the main satellite and its subsatellites.

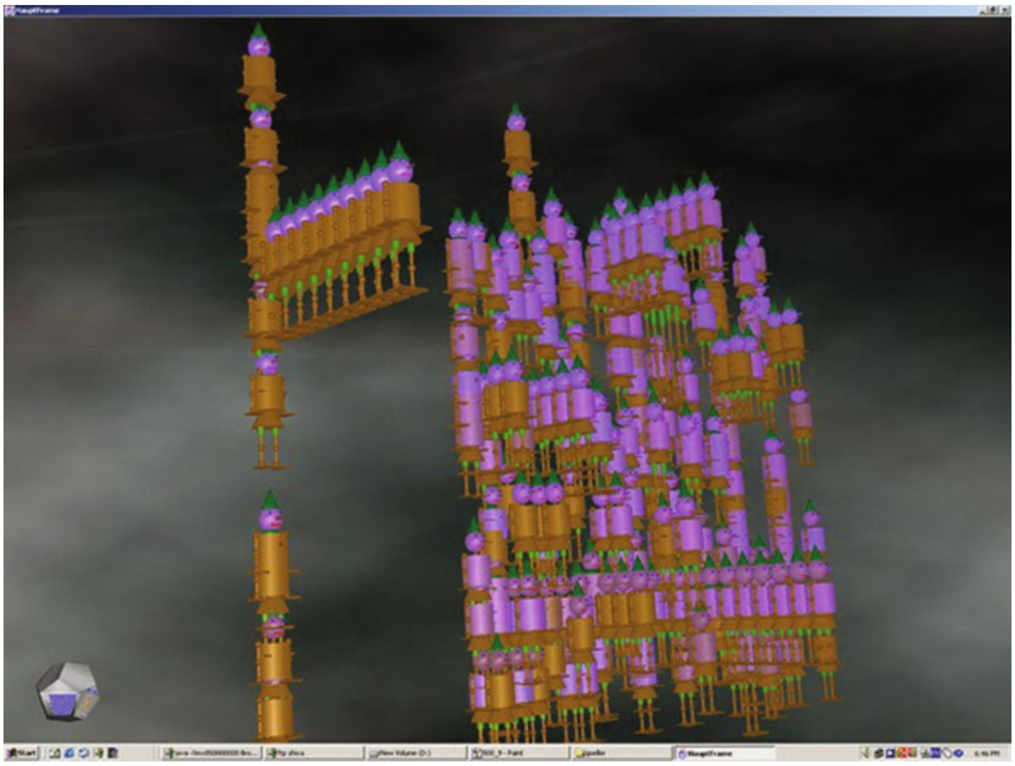


Fig. 71.2. A denotator visualized in *PrimaVista* using Pinocchios (satellites) of varying size and differently positioned extremities (subsatellites).

Second, several simplifications of the denotator concept were made to enable representation within this model. Göller does not, for instance, consider higher-dimensional **Simple** forms, such as ones using modules based on $\mathbb{R}^2$ or $\mathbb{C}$. Third, he mainly visualizes denotators on the topmost level, thereby assuming that it consists of a **Power** [372, p.63]. The BigBang rubette offers solutions to several of these problems, as discussed later.

71.1.2 Milmeister's ScorePlay and Select2D Rubettes

Even though the focus of Milmeister's work lay in building the basic mathematical framework as well as the interface of *RUBATO*® *Composer*, some of his rubettes offer visualizations of denotators of both general and specific nature. The *ScorePlay* rubette limits itself to *Score* denotators and represents them in piano roll notation. It simply visualizes a *Score* and enables users to play it back at a variable tempo and using different built-in MIDI instruments. It does not allow for any interaction with the represented notes.

The *Select2D* rubette represents any incoming **Power** or **List** denotator as points projected to a customizable two-dimensional coordinate system, the axes of which can be freely associated with any **Simple** denotator somewhere in the denotator hierarchy. Users can then select any number of these points by defining polygons around them (Figure 71.3). The rubette then outputs the subdenotators associated with these points as one runs the network.

Milmeister's rubettes provide several improvements over Göller's software while being more limited in other ways. *ScorePlay* only accepts denotators of one form and visualizes them rigidly. However, its visualization is minimal and based on a standard immediately understandable by the user, which Göller's might not always be. *Select2D*, in addition to **Power** denotators, also accepts **List** denotators, which were only introduced in Milmeister's work [739, p.105]. Furthermore, it is able to represent more types of **Simple** denotators than Göller's, more precisely ones containing free modules over any number ring except for $\mathbb{C}$. Nevertheless, higher-dimensional **Simple** coordinates and product rings can again not be represented.

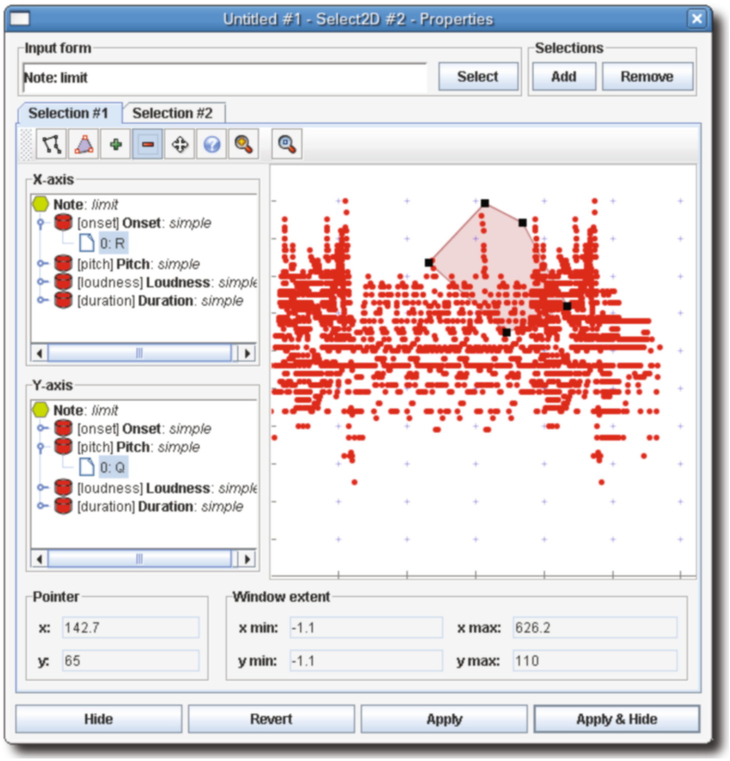


Fig. 71.3. The *Select2D* rubette showing a *Score* denotator on the *Onset* × *Pitch* plane.

Furthermore, the rubette's visualization capabilities do not exceed the representation of points projected to a two-dimensional coordinate system.

71.2 An Early Score-Based Version of BigBang

Initially, the *BigBang* rubette was designed for a small set of score-related denotators. The first version allowed users to handle *Scores* and *MacroScores* and was developed before in the context of an independent research project at the University of Minnesota [1043]. *MacroScore* is a conceptual extension of the form *Score* which we casually defined earlier. It brings hierarchical relationships to *Notes* by imitating the set-theoretical concept of subsets.[3] The form is defined in a circular way, as follows:

$$\begin{aligned}
&MacroScore : .\mathbf{Power}(Node),\\
&Node : .\mathbf{Limit}(Note, MacroScore),\\
&Note : .\mathbf{Limit}(Onset, Pitch, Loudness, Duration, Voice)
\end{aligned}$$

Each *Node* associates thus a *Note* with a set of again *Nodes*, each of which again contains a *Note* and a set, and so on. In short, with this construction, each *Note* of a *MacroScore* has a set of so-called *satellites* on a lower hierarchical level. We could go on infinitely, but in order to stop at some point, we give some of the *Nodes* empty sets, thus no satellites. The idea behind this form is that in music, we not only often group objects together and wish to treat them as a unity, but also establish hierarchies between them. A trill, for

[3] This complies with Graeser's notion of counterpoint as "a set of sets of sets of notes", cited in Section 13.1.

instance, consists of a main note, enhanced by some ornamental subnotes.[4] A simplified trill denotator could be defined as follows:

$$\begin{aligned}
&shakeWithTurn : @MacroScore(mainNode),\\
&mainNode : @Node(mainNote, ornamentalNotes),\\
&mainNote : @Note(\ldots),\\
&ornamentalNotes : @MacroScore(\\
&\quad upNode, midNode, upNode, midNode, lowNode, midNode),\\
&upNode : @Node(upNote, emptySet),\\
&upNote : @Note(\ldots),\\
&emptySet : @MacroScore(),\\
&\ldots
\end{aligned}$$

What is crucial to the notion of satellites is that their values are defined relatively to the ones of their anchor. So if for instance the *mainNote* defined above has *Pitch* 60 and its satellite *upNote Pitch* 61, the latter in fact obtains a *Pitch* of 1. If another had *Pitch* 58 it would be defined as -2. This way, if we transform the anchor, all its satellites keep their relative positions to it.

Later on, another form was added to *BigBang*'s vocabulary, *SoundScore*, which combines frequency modulation synthesis with the *MacroScore* concept. Each note, in addition to having satellites, can have modulators which modulate its frequency and change its timbre [1045]. Again, modulators have a relative position to their carrier and would be transformed with it. The form is defined as follows:

$$\begin{aligned}
&SoundScore : .\mathbf{Power}(SoundNode),\\
&SoundNode : .\mathbf{Limit}(SoundNote, SoundScore),\\
&SoundNote : .\mathbf{Limit}(Onset, Pitch, Loudness, Duration, Voice, Modulators),\\
&Modulators : .\mathbf{Power}(SoundNote)
\end{aligned}$$

Denotators of these forms are all based on the same five-dimensional space spanned by the **Simple** forms *Onset*, *Pitch*, *Loudness*, *Duration*, and *Voice* and can thus be visualized the same way. The early *BigBang* rubette did this using a generalized piano roll representation, as we will explain later on.[5] In sum, all of the objects the early *BigBang* rubette dealt with were essentially notes.

71.2.1 The Early BigBang Rubette's View Configurations

The visualization principle of the BigBang rubette [1043, p.4-5] combines elements of both Göller's and Milmeister's models, but focuses on a minimalist appearance aiming towards simplicity and clarity. It generalizes the piano roll notation also used in the ScorePlay rubette (see Section 71.1.2). Notes are represented by rectangles on a two-dimensional plane, just as in a piano roll. However, already in early versions of BigBang, the visual elements of the piano roll were separated from their original function so that they could be arbitrarily assigned to the symbolic dimensions of the represented score denotator. This is reminiscent of the ways Göller's subsatellites could be assigned to any folded denotator dimensions (Section 71.1.1) or of the spacial representation of Milmeister's *Select2D* rubette (Section 71.1.2). A similar method of visualizing was also available in *presto*®'s local views (see Section 50).

In order to do this we defined a set of six visual parameters

$$N = \{X\text{-}Position, Y\text{-}Position, Width, Height, Opacity, Color\}$$

[4] In a similar way, Schenkerian analysis describes background harmonic progressions enhanced by ornamental foreground progressions, which could be represented with *MacroScores* as well. However, we may find forms that are better suited, as will be discussed below.

[5] Piano roll is a standard in music software.

corresponding to the visual properties of piano roll rectangles, along with a set of six note parameters

$$M' = \{Onset, Pitch, Loudness, Duration, Voice, SatelliteLevel\},$$

which corresponds to the **Simple** denotator in *Scores* with the exception of *SatelliteLevel*, which was used to capture the hierarchical level of satellite notes in *MacroScores* and *SoundScores*. We then defined a *view configuration* to be a functional graph $V \subset N \times M'$. This ensures that each screen parameter $n \in N$ is associated with at most one musical parameter $V(n)$ that defines its value, as well as that V does not need to include all $n \in N$. View parameters not covered by V obtain a default value that can be defined by the user. The traditional piano roll notation could be produced by selecting the following pairing (shown in Figure 71.4):

$$V_1 = \{(X\text{-}Position, Onset), (Y\text{-}Position, Pitch), (Width, Duration)\}.$$

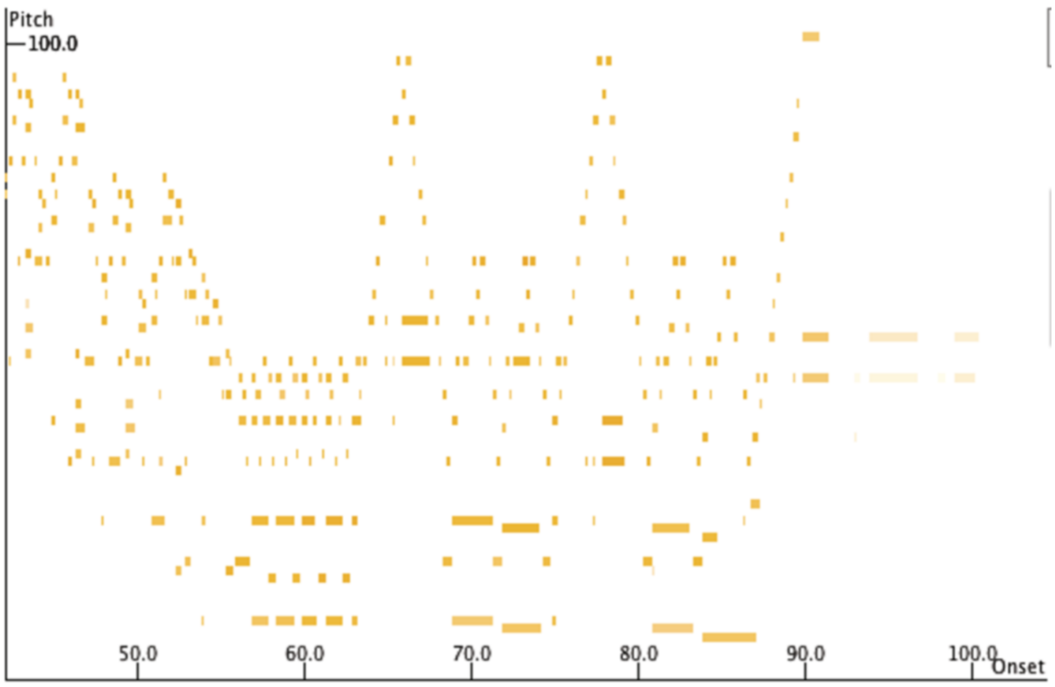


Fig. 71.4. The early *BigBang* rubette showing a *Score* in piano roll notation.

An enhanced version of the piano roll that often appears in software products also uses opacity and color:

$$V_2 = \{(X\text{-}Position, Onset), (Y\text{-}Position, Pitch), (Opacity, Loudness), (Width, Duration), (Color, Voice)\}.$$

The possibility of arbitrary pairings, however, also enables more adventurous but possibly also interesting view configurations, such as the following (Figure 71.5):

$$V_3 = \{(X\text{-}Position, Onset), (Y\text{-}Position, Loudness), (Width, Pitch), (Color, Onset), (Height, Loudness)\}.$$

Experimenting with such view configurations may be especially valuable for analysis and may lead to a different understanding of given musical data sets.

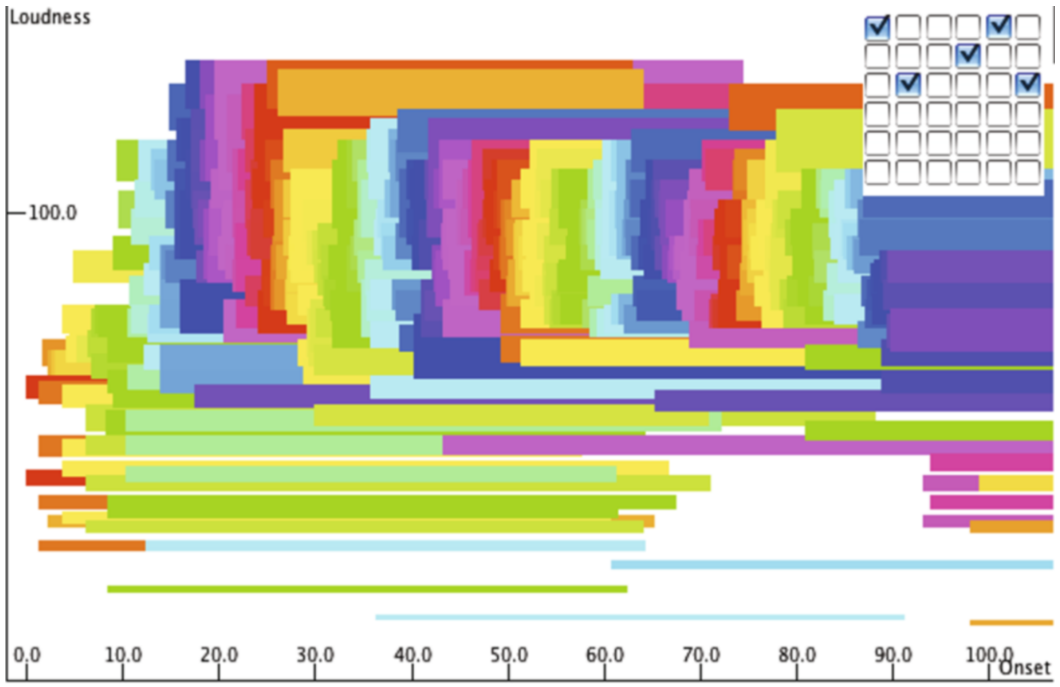


Fig. 71.5. The early *BigBang* rubette visualizing a *Score* in a more experimental way.

Every view parameter can be customized at runtime. Depending on the represented note parameter, it can be useful to ensure that a screen parameter's value does not exceed a specific value range. For example it may look more clear when the rectangle's heights are limited in a way that their areas do not intersect, just as with piano roll notation. Thus, for each $n \in N$, we optionally define min_n and max_n, the minimal and maximal screen values. We then have two options to define the way note parameters are mapped to the screen parameters.

1. If we choose the conversion to be *relative*, the minimal and maximal values of the given note parameters $min_m, max_m, m \in M'$, are determined for the actual score, and then mapped proportionally so that the note with min_m is represented by min_n and the note with max_m by max_n. For this, we use the formula

$$v_n = \frac{v_m - min_m}{max_m - min_m}(max_n - min_n) + min_n,$$

where v_n is the screen value for the note value v_m.

2. On the other hand, *absolute* mapping means that every value with $v_m < min_n$ or $v_m > max_n$ is mapped to a new value, while all other values stay the same, i.e., $v_n = v_m$. For absolute mapping, we have two choices. In *limited* mapping, the values that surpass the limits are given the min_n and max_n values, respectively. The following formula is used:

$$v_n = \begin{cases} min_n, \text{ if } v_m < min_n \\ max_n, \text{ if } v_m > max_n \\ v_m \text{ otherwise.} \end{cases}$$

For *cyclical* mapping, we use the formula

$$v_n = \begin{cases} (v_m \mod (max_n - min_n)) + min_n, \\ \qquad \text{if } v_m < min_n \text{ or } v_m > max_n \\ v_m \text{ otherwise.} \end{cases}$$

This mapping type can be useful for the color screen parameter for example, where it is reasonable to cycle through the color circle repeatedly to visualize a specific note parameter interval, such as an octave in pitch, or a temporal unit, as shown in Figure 71.5, where color visualizes a time interval of length 24, i.e., six 4/4 measures.
With absolute mapping it is possible to leave either or both of the **Limit**s as undefined. Accordingly, we assume $min_n = -\infty$ or $max_n = \infty$. Of course, if none of the limits are defined, the visible screen parameters correspond exactly with the original note parameters.

At runtime, the view window's current pairings could be selected using a *matrix of checkboxes* with a column for each screen parameter and a row for each note parameter, see Figure 71.5.

Satellite relations can be displayed in two ways. First, the note parameter *SatelliteLevel*, mentioned above, can be assigned to any arbitrary visual parameter. This way, anchor notes are associated with integer value 0, first-level satellites with 1, and so on. On the other hand, satellite relations may also be displayed as lines between the centers of two note objects so that every note has lines leading to each of its direct satellites, as shown in Figure 71.6.[6] As mentioned above, since all anchors and satellites in *MacroScore* and *SoundScore* denotators are notes, they can be represented in the same space.

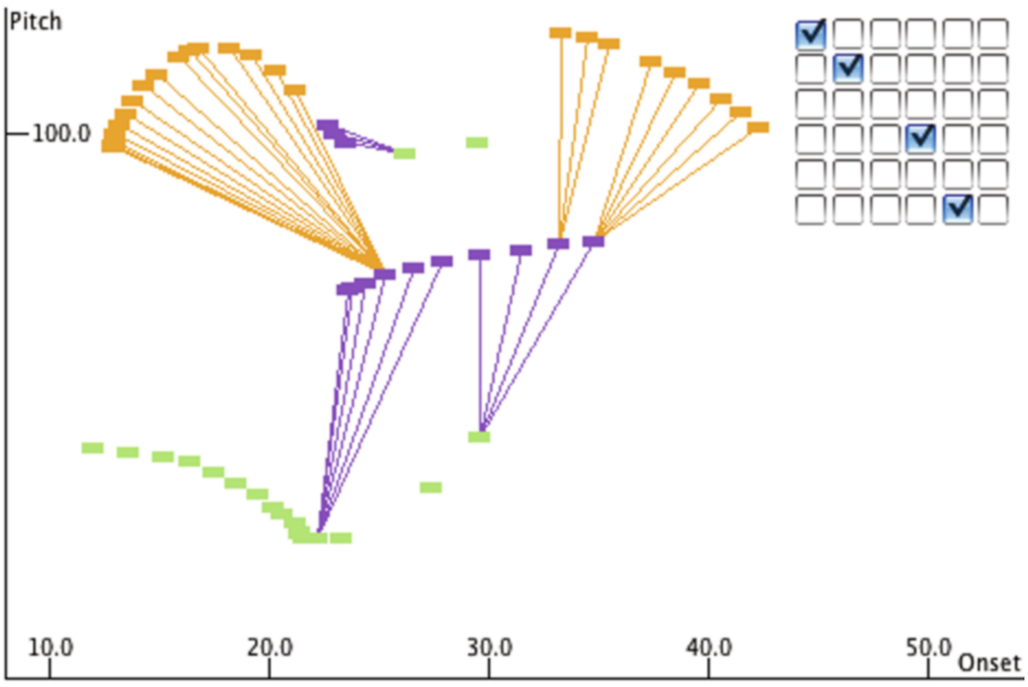


Fig. 71.6. The early *BigBang* rubette showing a *MacroScore* with two levels of satellites.

[6] This is a notion of satellites significantly different from Göller's (Section 71.1.1). While Goeller uses the term to denote movable parts of objects and represents them as denotators, but here we use it to speak of circular denotator structures (also see Note 2).

71.2.2 Navigating Denotators

Users can navigate this two-dimensional space not only by changing their view of the space by choosing different note parameters for the x and y view parameters, but also by changing their viewpoint by scrolling the surface and zooming in and out without limitations. This is similar to Göller's *PrimaVista*, but using two instead of three dimensions. However, users can also open several of these views simultaneously and choose different perspectives on the composition. This is especially valuable when performing transformations in one view while observing how the composition is affected from the other perspective.

71.2.3 Sonifying Score-Based Denotators

In early *BigBang*, denotators could not only be visualized but also sonified. Even though this may be done using another, specialized rubette such as *ScorePlay*, we decided to include this functionality within *BigBang*. The main reason for this was the gestural interaction concept, where immediate auditive feedback is key, in order to evoke a sense of continuity of motion. Users have to be able to judge musical structures by ear while they are creating them, and the use of an external rubette would have slowed down the process. A second reason was that many of the possible musical structures in early *BigBang* were micro-tonal, for which MIDI feedback, as implemented in *ScorePlay*, is unsuited since it is strictly chromatic.[7] The extension of *BigBang* for *SoundScores* was another reason, for now timbre was part of the musical objects and had to be judged while it was defined.

Since all the structures dealt with in early *BigBang* were *Score*-based denotators, sonification was rather straightforward. All the objects that had to be played were *Notes* that existed in the same space. They were simply played back in time, giving the user control over tempo. The microtonal and frequency modulation structures of *SoundScores* made it necessary for a synthesizer to be used. For each note, a synthesizer object, a so-called `JSynNote` footnoteJSyn is the name of the synthesizer framework we decided to use, as will be explained later on. was created by converting symbolic time, pitch, and loudness into the physical parameters time, frequency, and amplitude.

Outside *BigBang*, *MacroScores* usually have to be converted into *Scores* in order to be played back, a process called *flatten* (see next chapter). In early *BigBang*, this happened in the background, since it would have significantly slowed down the composition process. Satellites were simply converted into additional `JSynNotes` accordingly. Modulators in *SoundScores*, however, became modulators of `JSynNotes`. There were two options for playing back modulators: either their temporal parameters were ignored and they simply played whenever their carrier was playing, or they only modulated their carrier according to their own onset and duration. In the latter case, users had to make sure the anchor notes were playing at the same time as their modulators, but they also had the chance to create temporally varying configurations of modulators for a single note.

71.3 BigBangObjects and Visualization of Arbitrary $Mod^{@}$ Denotators

Despite its customizability, the view concept of the early *BigBang* rubette was first designed to represent *Score*, *MacroScore*, and *SoundScore* denotators, which are all based on the same musical space: (**Power** of...) **Power** of **Limit**. There, the view concept has proven its viability, compared to other concepts such as the ones discussed above. Now how can this concept be generalized so that *BigBang* can accept any denotator?

In this section we describe how we can do this for a major part of the spaces available in Milmeister's version of RUBATO® Composer, which are all based on elements of the topos $Mod^{@}$ over the category of modules, as described earlier.[8] The number of denotator types capable of being represented by the new

[7] The use of pitch bend is an option for monophonic material, but limited as soon as several notes have to be bent in different ways.

[8] This procedure is also described in [1048, p.3], as well as briefly in [1047].

BigBang rubette is significantly higher than the two comparable modules *PrimaVista* and *Select2D*. Nevertheless, for the time being we restrict ourselves to 0-addressed denotators and focus on number-based modules. We exclude both modules based on polynomial rings and ones based on string rings, since their visualization may differ markedly and will be left to future projects.

71.3.1 A Look at Potential Visual Characteristics of Form Types

As a starting point we need to reflect on the role of the five form types **Simple**, **Limit**, **Colimit**, **Power**, and **List** and the way they can best be visualized. Each of these types implies another visual quality that may be combined with the others. These qualities in early *BigBang Scores* were shown as clusters of rectangles (**Power**) within a coordinate system (**Limit**) of five axes (**Simple**), which could in turn be variably shown as any of six visual dimensions (x-position, y-position, width, height, color, opacity). Three of the five form types are involved here. The **Simple**s in a *Note* are based on free modules on a number ring and can thus easily be represented by one number axis or one of the other visual properties. However, *Rubato Composer* allows for many more types of **Simple**s, each of which must be considered here.

71.3.1.1 Simple Denotators

Simple denotators are crucial to a system of visualization, since they are the only denotators that stand for a specific numerical value in a space. Basically, every form that will be used in a practical way should contain **Simple**s. This despite the fact that it is possible to conceive more pathological forms, such as the circular form that describes sets as sets of sets:

$$Set : .\mathbf{Power}(Set).$$

Such forms will be of little use in our context, since anything to be represented and especially transformed needs to contain specific numerical values. We can thus declare a first rule here:

Rule 1 *In our system denotators will only be represented if they contain at least one* **Simple** *denotator somewhere in their structure.*

With the system, **Simple** denotators over the following modules can be represented:

Free Modules over Number Rings

The most straightforward type of modules are the free modules based on number rings such as $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, or $\mathbb{C}$. Elements of the first three are typically represented along an axis, whereas ones of the last on a two-dimensional Cartesian system. For modules $\mathbb{Z}^n, \mathbb{Q}^n, \mathbb{R}^n$, and $\mathbb{C}^n$ an n-dimensional or $2n$-dimensional system of real axes will be appropriate.

Furthermore, as shown in Section 71.2.1, as long as all values of a specific denotator are known and finite, dimensions of free modules over number rings can equally be represented by other visual parameters, such as an object's width, height, color, etc. Elements of the free module over $\mathbb{C}$, for instance, could convincingly be represented as width and height of objects.

Quotient Modules

For free modules over quotient modules such as $\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z}$, $\mathbb{Q}_m = \mathbb{Q}/m\mathbb{Z}$, $\mathbb{R}_m = \mathbb{R}/m\mathbb{Z}$, and $\mathbb{C}_m = \mathbb{C}/m\mathbb{Z}$ we choose a manner of representation that corresponds to the one introduced in the previous section, where values are simply projected on a one- or two-dimensional coordinate system. However, instead of being potentially infinite, the axes maximally show the numbers of the interval $[0, m[$, which makes zooming out beyond this point impossible. This works in an analogous way for other view parameters that do not allow cyclical representation, such as width, height, and opacity. Of the defined visual parameters, only color allows for cyclical representation, in analogy to the color circle. Again, for $\mathbb{Z}_m^n, \mathbb{Q}_m^n, \mathbb{R}_m^n$, and $\mathbb{C}_m^n$, the system can be extended to be n-dimensional or $2n$-dimensional.

Modules over Product Rings and Direct Sums of Modules over Quotient Modules

Representation is straightforward for direct sums of any of the quotient modules discussed so far. Each of the factors is independently associated with one of the view parameters. For example, for $\mathbb{Z} \times \mathbb{R}_7$ we might choose to represent the $\mathbb{Z}$ part with the x-axis and the $\mathbb{R}_7$ part with color.

Remark: More general modules which are not derived from direct sums of such quotient modules are not yet dealt with.

71.3.1.2 Limit Denotators

The fact that **Limit**s are products or conjunctions makes them always representable in the conjoined space, i.e., the cartesian product of the spaces of their factors. The most simple case is a **Limit** of **Simple** denotators, just as with our common *Score* denotators. *Notes* can be represented in $Onset \times Pitch \times Loudness \times Duration \times Voice$. This is even possible if the same subspaces appear in several times. For instance, if we define a form

$$Dyad : .Limit(Pitch, Pitch),$$

its denotators are representable in the space $Pitch \times Pitch$. This also works in cases where the factors of a **Limit** are not directly **Simple**.

71.3.1.3 Colimit Denotators

Colimits, disjunctions or coproducts, are again representable in the product space of their cofactors, even if they then typically do not have defined values in all of the product's dimensions. For "missing" dimensions, we set standard values, so that the denotators are represented on a hyperplane in the entire product space. The case where cofactors share common subspaces is especially interesting, since these subspaces will always be populated.

An example will clarify this: we assume a form *EulerScore* which consists of *EulerNotes* and *Rests*, which share the **Simple** forms *Onset* and *Duration*. The product space of all cofactor spaces is $Onset \times EulerPitch \times Loudness \times Duration$. While $EulerNotes$ fill the entire space, *Rests* are simply represented on the $Onset \times Duration$ plane. Thus, even though $EulerNotes$ and *Rests* are actually separated by a coproduct, both can be shown in the same space.

71.3.1.4 Power and List Denotators

Power and **List** forms define sets and ordered sets of distinct objects on any hierarchical level. In practice we typically encounter them on the topmost level as for instance with all the forms supported by early *BigBang*, *Score*, $MacroScore$, and *SoundScore*. However, it is also conceivable that they occur only at lower levels, as in Mariana Montiel's more detailed score form, which is defined as [760]

$$Score' : .\mathbf{Limit}(BibInfo, Signatures, Tempi, Lines, GeneralNotes,\\ GroupArticulations, Dynamics).$$

There, **Power**s appear in almost all the coordinator forms, but not at the top level. In this case we can for instance see all *BarLine* or *Slur* denotators that appear on lower levels as indirect satellites of our main $Score'$.

Power denotators can always be represented as a set of points in the space of their coordinate. An *EulerScore*, for instance, can be shown as a cloud of objects in the $EulerNoteOrRest$ space described above. **List** denotators can be shown the same way, however, at the expense of the order of their elements, for it may contradict the spatial organization. In any case, **Power** and **List** forms are in fact the main constructs that define the discrimination of distinct visual objects. Wherever they occur, we have the opportunity to define as many elements as we would like.

71.3.2 From a General View Concept to BigBang Objects

From these characteristics we can imply that all we need to have for a representation of any denotators is a conjunction of the **Simple** spaces and a visualization of clouds of objects within them. These objects can be represented in just the same way as the ones in the generalized piano roll described above, as multidimensional rectangles. Whenever a denotator enters *BigBang*, the visualization space is reset based on its form, and users have the possibility to select any form space and start drawing objects, as will be described below.

We arrive at the core part of our generalization. In short, the representation of any arbitrary denotator relies on the fundamental difference between the various types of compound forms, **Limit**, **Colimit**, and **Power**. We propose a novel system of classification that generalizes the previous notion of anchors and satellites, based on occurrences of **Power** denotators. For this, we maintain the following rules:

Rule 2 The general visualization space consists of the cartesian product of all **Simple** form spaces appearing anywhere in the anatomy of the given form. For instance, if we obtain a $MacroScore$ denotator of any hierarchical depth, this is $Onset \times Pitch \times Loudness \times Duration \times Voice$.

Rule 3 Any **Simple** form X the module of which has dimension $n > 1$ is broken up into its one-dimensional factors $X_1, \ldots, X_n$. The visual axes are named after the dimension they represent, i.e. X_n, or X if $n = 1$.

Rule 4 If the same **Simple** form occurs several times in a **Limit**, it is taken to occur several times in the product as well. For instance, the product space of *Dyad* is $Pitch \times Pitch$. However, if the same **Simple** form occurs at different positions in a **Colimit**, this is not the case. For instance, **Colimit** of *Pitch* and *Pitch* results in the space *Pitch*. This renders the space more simple, but we also lose some information. This loss can be regained thanks to an additional spatial dimension, *cofactor index*, as described under Rule 7.

Rule 5 **Power** or **List** denotators anywhere in the anatomy define an instantiation of distinct visual objects represented in the conjoined space. Objects at a deeper level, i.e., contained in a subordinate **Power** or **List**, are considered *satellites* of the higher-level object and their relationship is visually represented by a connecting line. For example, $SoundScore$ objects formerly considered modulators are now visually no different from regular satellites.

Rule 6 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. However, if an object is a satellite and one of the **Simple** forms associated with the axes occurs anywhere in its parental hierarchy, it is represented at exactly that value.

Rule 7 If there is an occurrence of either **Colimit**s or *satellites*, additional dimensions are added to the ones defined in Rules 1-3. For **Colimit** we add *cofactor index*, and for satellites *sibling index* and *satellite level*.[9] These dimensions are calculated for each object and can be visualized in the same manner as the other ones. For instance, associating satellite level with y-position facilitates the selection of all denotators at distinct positions of the satellite hierarchy.

We call the objects defined by these rules `BigBangObjects`. They are not only visual entities, but they are the entities that the *BigBang* rubette deals with in every respect. All operations and transformations available in *BigBang* are applied to sets of `BigBangObjects`, as we will see in the next chapter. The consequence is that we simplify the structure of forms and denotators significantly, so that if we, for instance, are handling denotators of a form defined as **Limit** of **Limit** of **Limit** and so on, we can treat it as a single object. New objects are broken up only if there are **Power**s or **List**s in the hierarchy. We can thus for instance claim that in *BigBang* we assume that

$$\mathbf{Limit}(A, \mathbf{Limit}(B, \mathbf{Limit}(C, D))) = \mathbf{Limit}(A, B, C, D).$$

71.3.2.1 Implications for Satellites

One of the main innovations of these definitions is a new notion of the concept of satellites. Previously, the term was uniquely used to describe $Notes$ in a $MacroScore$ that are hierarchically dependent on other

[9] Already present in the early *BigBang*, as seen in Section 71.2.1.

Notes. For instance, the analogous construction of *Modulators* in *SoundScores* was not referred to in this way; neither was the relationship represented the same way as satellites are [1045]. Following Rule 5 above, *Modulators* are now equally considered satellites and represented in precisely the same way. Another new aspect of this is that now satellites do not technically have to have a shared space with their anchor. For instance, if we define

$$\begin{aligned}&HarmonicSpectrum : .\mathbf{Limit}(Pitch, Overtones),\\&Overtones : .\mathbf{Power}(Overtone),\\&Overtone : .\mathbf{Limit}(OvertoneIndex, Loudness),\\&OvertoneIndex : .\mathbf{Simple}(\mathbb{Z}),\end{aligned}$$

Overtones do not have a *Pitch* themselves, but merely a *Loudness*. Because of Rule 6, however, if we choose to see *Loudness* × *Pitch* as the axes of a view configuration, the *Overtones* are represented above their anchor. An example visualization of this form will be shown below.

Above, we discussed how satellites and modulators were defined relatively to their anchors (Section 71.2). This can also be generalized for the new notion of satellites. We add another rule:

Rule 8 Given a **Simple** form F, every denotator $d_i : @F$ in a satellite `BigBangObject`, i being its index in case the satellite contains several denotators of form F, is defined in a relative way to $d_i@F$ in its anchor, if there is such a denotator.

For instance, if we define

$$\begin{aligned}&MacroDyad : .\mathbf{Power}(DyadNode),\\&DyadNode : .\mathbf{Limit}(Dyad, MacroDyad),\end{aligned}$$

the first *Pitch* in each satellite *Dyad* is defined relatively to the first *Pitch* in its anchor, and the second *Pitch* in each satellite relatively to the second *Pitch* in the anchor. On the other hand, in a *HarmonicSpectrum* none of the satellites share **Simple** denotators with their anchor and are thus defined absolutely.

71.3.3 New Visual Dimensions

The facts view of the new *BigBang* maintains all the features of the early *BigBang* and can still be navigated the same way as described in Section 71.2.2. However, the newest version allows independent zooming in and out horizontally and vertically, when the shift or alt keys are pressed. It also features some additional view parameters. There is now an option to use, instead of hue (*Color*) values, RGB values for color, in a similar way as that in *PrimaVista*. This adds more visual variety at the expense of the cyclical nature of hue, and is especially beneficial when working with data types other than musical ones, such as images. The new view parameters vector thus looks as follows:

$$N' = \{X\text{-}Position, Y\text{-}Position, Width, Height, Alpha, Red, Green, Blue\}.$$

In the future, more visual characteristics can easily be added, such as varying shapes, texture, or a third dimension.

The former note parameters, in turn, now called *denotator parameters*, include *SiblingNumber* and *ColimitIndex* where appropriate and vary according to the input or chosen form. The former identifies the index of a denotator in its **Power** or **List**, whereas the latter refers to an index based on all possible combinations of **Colimit** coordinates. For instance, for an object form

$$\mathbf{Colimit}(\mathbf{Colimit}(X_0, X_1), \mathbf{Colimit}(X_2, X_3, X_4), X_5),$$

where $X_0, ..., X_5$ are any other forms not containing **Colimit**s, we get six possible configurations: an object containing X_0 gets index 0, one containing X_1 gets 1, and so on.

For *EulerScore* denotators, for instance, the entire denotator parameters look as follows:

$$M_{EulerScore} = \{Onset, EulerPitch1, EulerPitch2, EulerPitch3, \\ Loudness, Duration, ColimitIndex\}.$$

The three-dimensional space of *EulerPitch* is broken up into its three constituent dimensions, and *ColimitIndex* is added, with two potential values: 0 for *EulerNoteOrRests* containing an *EulerNote*, 1 for *EulerNoteOrRests* with a *Rest*.

71.4 The Sonification of BigBangObjects

As seen above in Section 71.2.3, in the early *BigBang* rubette, sonification was relatively straightforward, since all objects that had to be dealt with existed in the same five-dimensional space. For the new *BigBang*, this concept had to be generalized as well. For users to be able to sonify a multitude of denotators, even ones they define themselves, the sonification system had to become more modular.

Our solution generalizes the `JSynNotes` described above into `JSynObjects`, which can contain any number of a set of standard musical parameters. Each `BigBangObject` is converted into a `JSynObject`, by searching for occurrences of these musical parameters anywhere in their anatomy. Any parameters necessary for a sounding result subsequently obtain a standard value. For instance, if we play back a **Simple** denotator *Pitch*, a `JSynObject` is created with a standard *Loudness*, *Onset*, and *Duration*, so that it is audible. Especially *Onset* and *Duration* are relevant in this case. The standard values assigned for temporal parameters are chosen such that the object plays continuously for as long as the denotator is being played. This is particularly interesting when the denotator is transformed, which results in continuously sounding microtonal sweeping.

`JSynObjects` can also have multiple pitches, in order to work with denotators such as *Dyad*, as defined in Section 71.3.1.2, or other user-defined types that might describe chords, and so on. Some of the recognized simple forms so far are all note parameters (*Onset*, *Pitch*, *Loudness*, *Duration*, *Voice*), as well as *BeatClass*, *ChromaticPitch*, *PitchClass*, *TriadQuality*, *OvertoneIndex*, *Rate*, *Pan*, and *OperationName*. *Rate* replaces *Onset* by defining the rate at which a `JSynObject` is repeatedly played, *OperationName* distinguishes between frequency modulation, ring modulation, and additive synthesis, and *TriadQuality* adds an appropriate triad above each *Pitch* in the object, assuming that they are root notes. Some of the other forms are discussed below, along with examples of the visualization of their denotators.

Another recent addition is the option of having everything played back through MIDI, either with Java's internal MIDI player, or by sending live MIDI data to any other application or device, via IAC bus or MIDI interface. MIDI is event-based and thus problematic for playing continuous objects without temporal parameters. There are two solutions to this problem implemented so far. Either, objects are repeated continuously at a specified rate, or a note-off event is only sent when an object is replaced by another. In the latter case, note ons are only sent again once a denotator is transformed.

In order to play back the composition in *BigBang*, users can press the play button in the lower toolbar. If the denotator has a temporal existence, i.e., it contains *Onsets*, it can be looped, where the player automatically determines the loop size to be the entire composition. In addition to this, any musical denotator in *BigBang* can be played back by using an external MIDI controller such as a keyboard controller. Each MIDI key of such a controller triggers one performance of the denotator, i.e., a one-shot temporal playback, a loop, or a continuous playback, depending on the denotator. Middle C (60) corresponds to the visible denotator, while all other keys trigger transposed versions, e.g. a half step up for 61, etc. This is especially practical when designing sounds, i.e., denotators without *Onset* or *Duration*, such as the *HarmonicSpectrum* form defined above. This way, users can design sounds and immediately play the keyboard with them, just as with a regular synthesizer.

For the future, this system of sonification could be extended in order to work in a similar way to view configurations. For now, whenever a new **Simple** form is introduced that should be sonified in a novel way, the system has to be adjusted accordingly. With a free association of any **Simple** form with a sonic parameter, just as is done for the visual system, users can experiment with spontaneously performing parameter exchanges, or with sonifying non-musical forms.

71.5 Examples of Forms and the Visualization of Their Denotators

In this chapter, we have discussed what the objects on *BigBang*'s factual level are and how they are visualized and sonified. It is now time to give some specific examples of forms that can potentially be defined and show how their denotators are visualized. Sonification will have to be left to the readers to try themselves. Anything we feed the new BigBang rubette will be analyzed and visualized as described above. Users may also select a form within *BigBang* upon which the facts view is cleared and they may simply start drawing denotators, as will be described in the next chapter.

We will start with some simple constructs from set theory, move to tonal constructs, and finally give some examples from computer music and sound design.

71.5.1 Some Set-Theoretical Structures

The most basic construct to be represented is necessarily a single **Simple** denotator. For instance, if we input a *Pitch*, the space is merely one-dimensional, but it can be represented in various visual dimensions simultaneously. Figure 71.7 shows the pitch $middleC : @Pitch(60)$ – C4 is MIDI pitch 60 – being represented in every possible visual dimension in RBG mode,[10] however reasonable this may be. X, y, width, height, alpha, red, green, and blue, all represent the value 60, depending on the min_m, max_m defined (see Section 71.2.1).

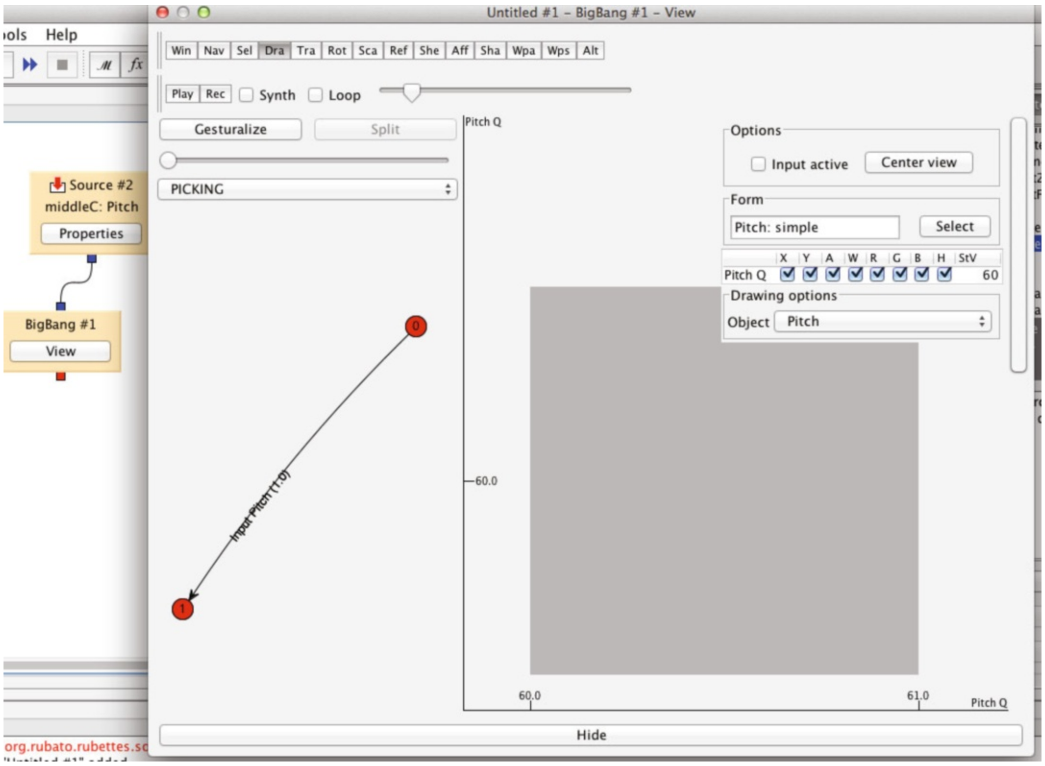


Fig. 71.7. The new *BigBang* rubette visualizing a *Pitch* denotator in every visual dimension.

[10] Explained in Section 71.3.3.

For a **Power** of a **Simple**, we get a cloud of values. Figure 71.8 shows an example of

$$\begin{aligned}&PitchSet : .\mathbf{Power}(ChromaticPitch),\\&ChromaticPitch : .\mathbf{Simple}(\mathbb{Z}).\end{aligned}$$

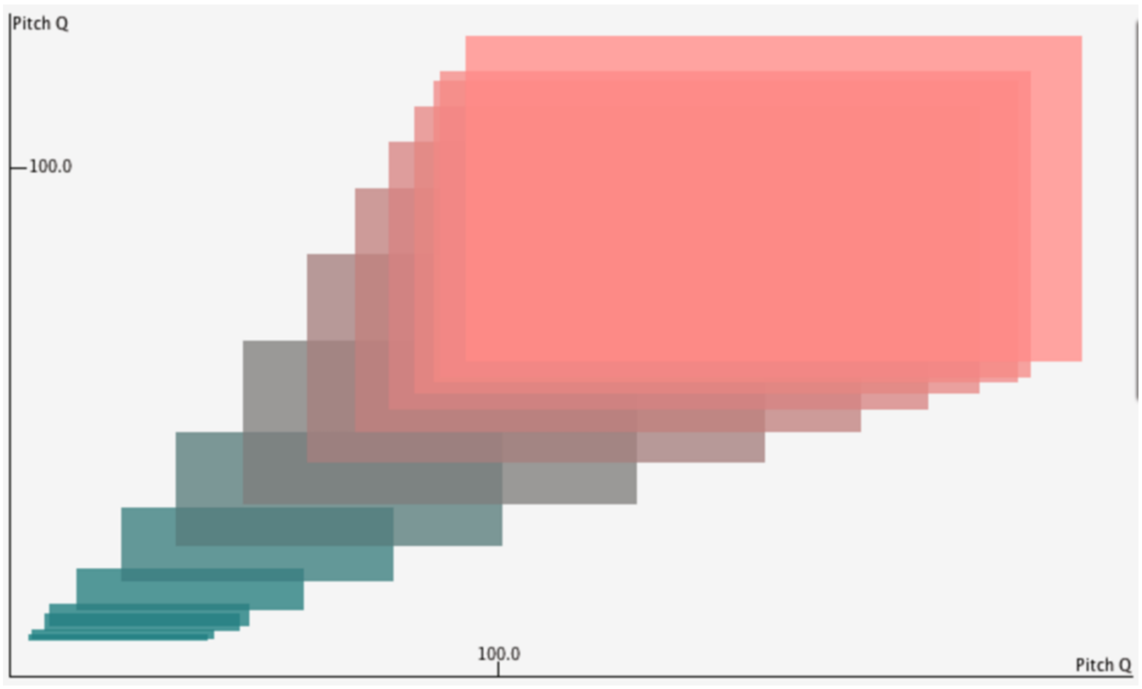


Fig. 71.8. A *PitchSet* simultaneously visualized using several visual characteristics.

Note that *ChromaticPitch* differs from *Pitch* in that it only allows for integer values, which models the Western equal-tempered chromatic pitch space. In the figure, *ChromaticPitch* is shown on both axes, color, width and height. This way, we can define all sorts of datatypes commonly used in music theory or sound synthesis and visualize and sonify them. If we wanted, for instance, to compose with pitch classes instead of pitches, we could define

$$\begin{aligned}&PitchClassSet : .\mathbf{Power}(PitchClass),\\&PitchClass : .Simple(\mathbb{Z}_{12}).\end{aligned}$$

If we wish to work with pitch-class trichords, a common construct in set theory, we can define

$$\begin{aligned}&Trichords : .\mathbf{Power}(Trichord),\\&Trichord : .\mathbf{Limit}(PitchClass, PitchClass, PitchClass).\end{aligned}$$

PitchSets and *PitchClassSets* can also be realized as ordered sets. We simply need to replace **Power** with **List**, e.g.

$$OrderedPitchSet : .\mathbf{List}(Pitch).$$

In order to compose with *PitchClasses* the same way we can compose with *Scores*, i.e., create temporal structures, we can define

$$\begin{aligned}&PitchClassScore : .\mathbf{Power}(PitchClassNote),\\&PitchClassNote : .\mathbf{Limit}(Onset, PitchClass, Loudness, Duration, Voice),\end{aligned}$$

which is then visualized as shown in Figure 71.9.

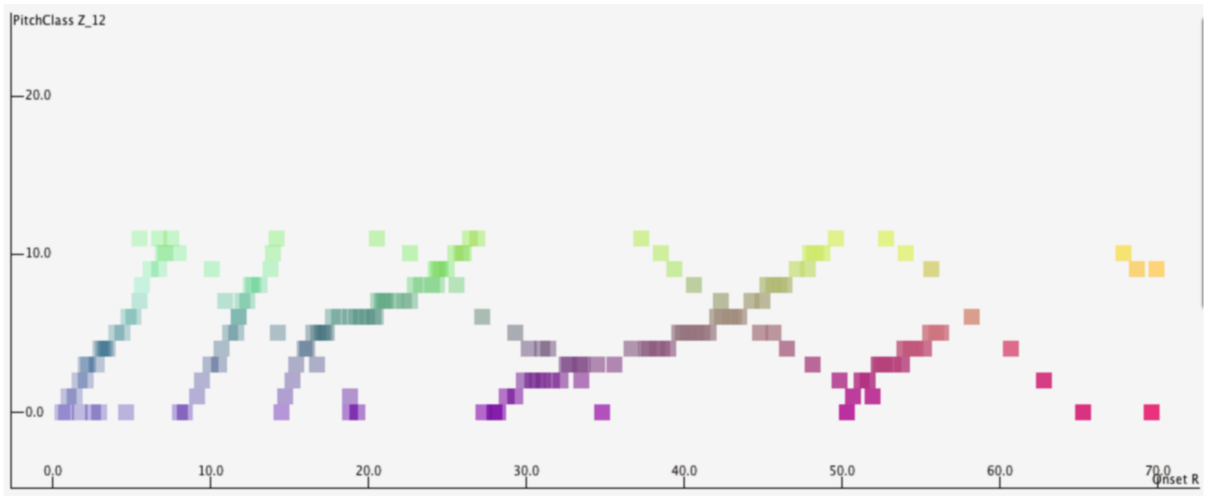


Fig. 71.9. A *PitchClassScore* drawn with ascending and descending lines to show the cyclicality of the space.

71.5.2 Tonal and Transformational Theory

The next few examples imitate spaces and constructs from transformational theory and traditional music theory.[11] For a model of triads, as they are often used in transformational theory, we define

$$\begin{aligned} Triad &: .\mathbf{Power}(Pitch, TriadQuality),\\ TriadQuality &: .\mathbf{Simple}(\mathbb{Z}_4), \end{aligned}$$

where Quality stands for one of the four standard qualities in tonal music: diminished, minor, major, and augmented.

More generally, a simplified notion of chord progressions can be implemented as follows:

$$\begin{aligned} Progression &: .\mathbf{List}(Chord),\\ Chord &: .\mathbf{Limit}(Onset, PitchSet, Loudness, Duration), \end{aligned}$$

assuming that all members of a chord have the same temporal and dynamic qualities. In so doing, the pitches of a chord are actually satellites and thus also visualized this way, as can be seen in Figure 71.10. From there, we can also define hierarchical chord progressions the same way as we did above for *Score* or *Dyad*. For instance, we can define

$$\begin{aligned} MacroProgression &: .\mathbf{List}(ChordNode),\\ ChordNode &: .\mathbf{Limit}(Chord, MacroProgression). \end{aligned}$$

This way, each chord can have ornamental progressions, just as we know it from Schenkerian theory. If a main progression is transposed, its ornamental progressions, defined in a relative way to them, are transposed with it. The next chapter will clarify what this means.

Figure 71.11 shows an example of the depiction of **Colimit**s. It shows a denotator of a form similar to *EulerScore*, but with regular *Pitch* and an additional *Voice* parameter, thus simply using regular *Notes* and *Rests*. In the image we see that all the rests are depicted at *Pitch* 0, since they do not contain a *Pitch*. If we chose to depict the denotator on the *Onset* $\times$ *Duration* plane, the rests would also be shown in two dimensions.

A final example illustrates a way we can introduce rhythmical relationships other than using *Onset*. If we write

[11] Some of them were described in [1048].

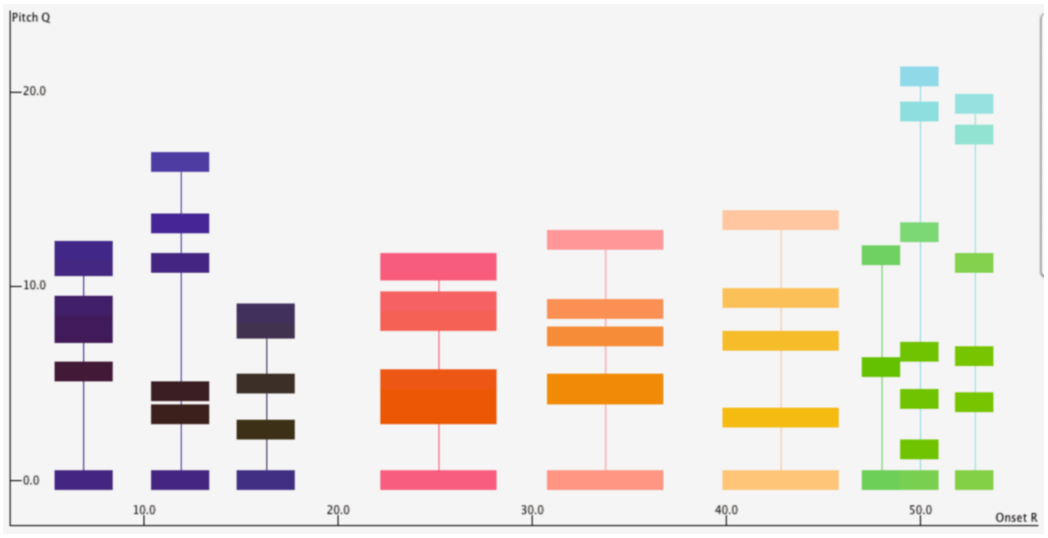
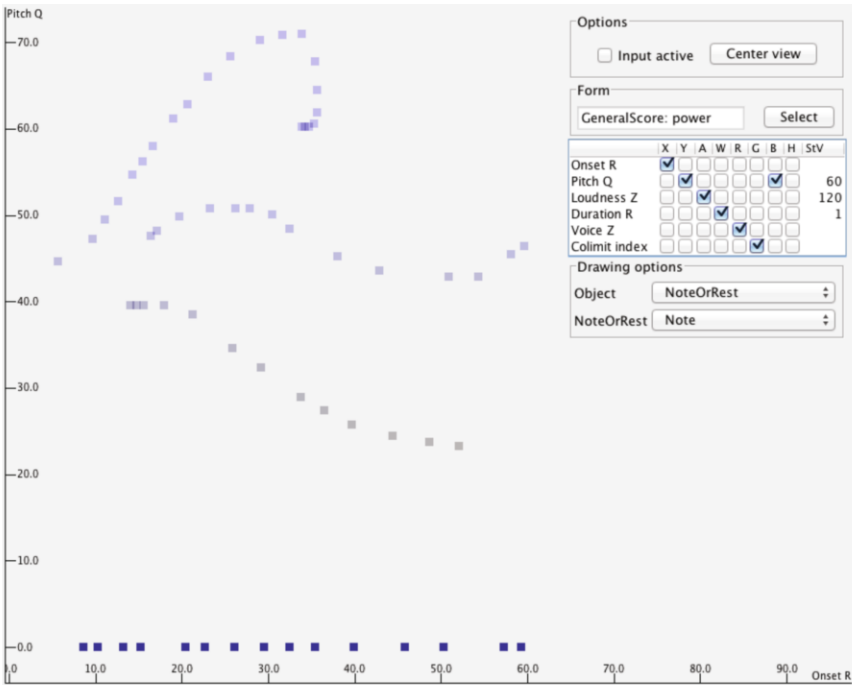


Fig. 71.10. A *Progression* where pitches adopt the visual characteristics of their anchor chord.



GeneralScore: power

	X	Y	A	W	R	G	B	H	StV
Onset R	✓								
Pitch Q		✓					✓		60
Loudness Z			✓						120
Duration R				✓					1
Voice Z					✓				
Colimit index						✓			

Object: NoteOrRest
NoteOrRest: Note

Fig. 71.11. A *GeneralScore* with some *Notes* and *Rests* shown on the *Onset* × *Pitch* plane.

$$
\begin{aligned}
&Texture : .\mathbf{Power}(RepeatedNote),\\
&RepeatedNote : .\mathbf{Limit}(Pitch, Loudness, Rate, Duration),\\
&Rate : .\mathbf{Simple}(\mathbb{R})
\end{aligned}
$$

we obtain a set of notes that are repeated at a certain rate, altogether forming a characteristic *Texture*.

71.5.3 Synthesizers and Sound Design

Finally, here are some examples of forms that allow for more sound- and timbre-oriented structures. Some of the forms shown in Section 71.5.1 could be considered to be sound-based forms as they may be seen as somewhat related to additive synthesis, but we can go much farther than that.[12]

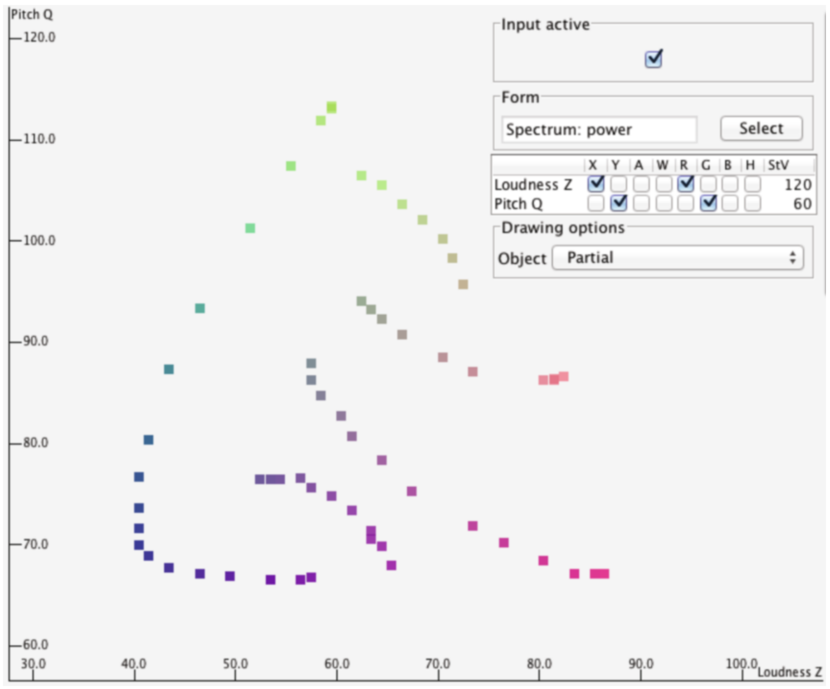
For instance, we can define

$$\begin{aligned} Spectrum &: .\mathbf{Power}(Partial), \\ Partial &: .\mathbf{Limit}(Loudness, Pitch). \end{aligned}$$

This models a constantly sounding cluster based on only two dimensions. Since it is not using *ChromaticPitch* but *Pitch*, the cluster can include any microtonal pitches. Figure 71.12 shows an example of a *Spectrum*. If we, however, wanted to define a spectrum that only allows for harmonic overtones, this form would not be well suited, as we would have to meticulously arrange each individual pitch so that it sits at a multiple of a base frequency. Instead, we could simply use the form already introduced above, *HarmonicSpectrum* (Section 71.3.2.1). Figure 71.13 shows an example denotator of a set of harmonic spectra, defined as

$$HarmonicSpectra : .\mathbf{Power}(HarmonicSpectrum).$$

Since satellites (*Overtones*) and anchors (*HarmonicSpectrum*) do not share **Simple** dimensions, they can only be visualized if one **Simple** of each is selected as an axis parameter, here $Pitch \times OvertoneIndex$. However, as we will see in the next chapter, they can both be transformed in arbitrary ways on such a plane. These are examples of the simplest way of working with additive synthesis in *BigBang*. All oscillators are expected to be based on the same wave form and a phase parameter is left out for simplicity. This is also the case for the following examples.



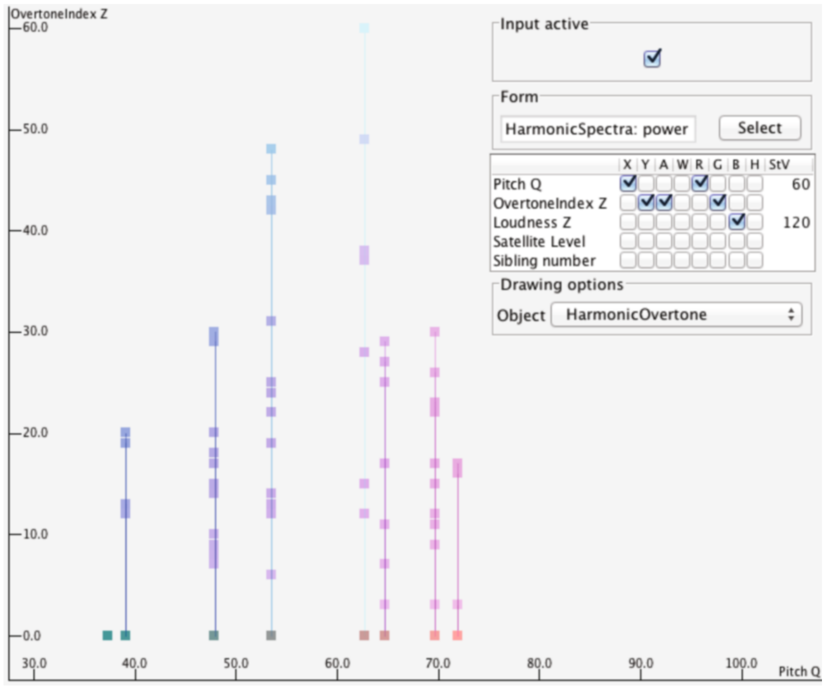
Form: Spectrum: power

	X	Y	A	W	R	G	B	H	StV
Loudness Z	✓				✓				120
Pitch Q		✓				✓			60

Object: Partial

Fig. 71.12. A *Spectrum* shown on $Loudness \times Pitch$.

[12] Some of these constructions were described in [1047].



Input active ✓

Form

HarmonicSpectra: power

Select

	X	Y	A	W	R	G	B	H	StV
Pitch Q	✓				✓				60
OvertoneIndex Z		✓	✓			✓			
Loudness Z							✓		120
Satellite Level									
Sibling number									

Drawing options

Object HarmonicOvertone

Fig. 71.13. A constellation of eight *HarmonicSpectra* with different fundamental *Pitches* and *Overtones*.

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 *Overtone* its own *Pitch*. The following definition does the trick:

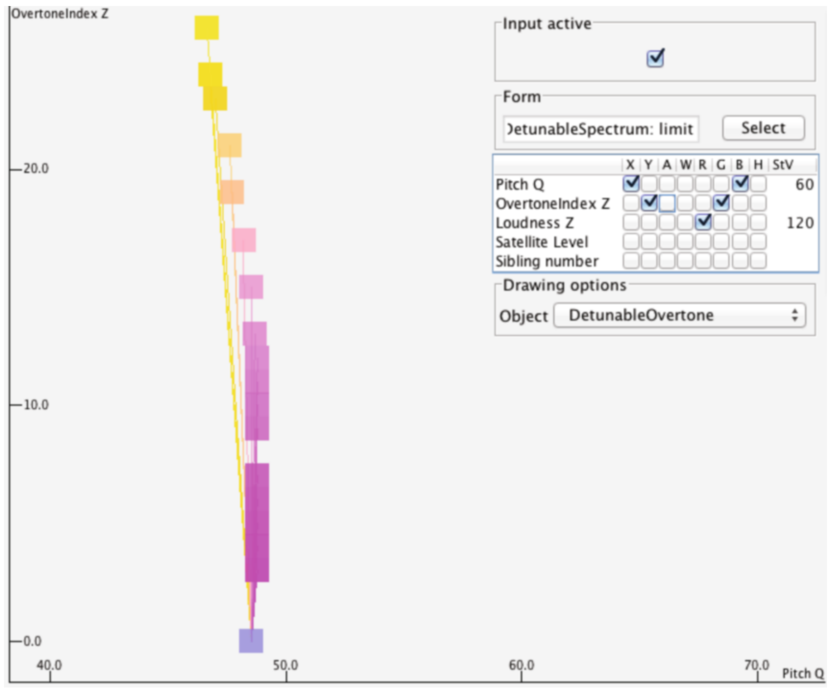
$$\begin{aligned}
&DetunableSpectrum : .\mathbf{Limit}(Pitch, Overtones),\\
&Overtones : .\mathbf{Power}(Overtone),\\
&Overtone : .\mathbf{Limit}(Pitch, OvertoneIndex, Loudness).
\end{aligned}$$

Since values reoccurring in satellites are defined in a relative way to the corresponding ones of their anchor, we get the opportunity to define *deviations* in frequency from the harmonic overtone, rather than the frequencies themselves. A displacement of a satellite on the *Pitch* axis with respect to its anchor enables us to detune them. Figure 71.14 shows an instance of such a *DetunableSpectrum*.

The three forms above are just a few examples of an infinite number of possible forms. 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 ways 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:

$$\begin{aligned}
&FMSet : .\mathbf{Power}(FMNode),\\
&FMNode : .\mathbf{Limit}(Partial, FMSet),
\end{aligned}$$

with *Partial* as defined above. Examples as complex as the one shown in Figure 71.15 can be created this way. Frequency modulation, typically considered highly unintuitive in terms of the relationship of structure and sound [199], can be better understood with a visual representation such as this one. All carriers and modulators are shown respective to their frequency and amplitude and can be transformed simultaneously and



Input active ✓

Form:)etunableSpectrum: limit Select

	X	Y	A	W	R	G	B	H	StV
Pitch Q	✓						✓		60
OvertoneIndex Z		✓				✓			
Loudness Z					✓				120
Satellite Level									
Sibling number									

Drawing options

Object: DetunableOvertone

Fig. 71.14. An instance of a *DetunableSpectrum*, where the fundamentals of the *Overtones* are slightly detuned.

parallelly, which has great advantages for sound design compared to old-fashioned skeuomorphic synthesizers and applications.

In order to include other synthesis models, we can define

$$\begin{aligned}
&GenericSound : .\mathbf{Limit}(Oscillator, Satellites, Operation),\\
&Oscillator : .\mathbf{Limit}(Loudness, Pitch, Waveform),\\
&Satellites : .\mathbf{List}(GenericSound),\\
&Operation : .\mathbf{Simple}(\mathbb{Z}_3),\\
&Waveform : .\mathbf{Simple}(\mathbb{Z}_4),
\end{aligned}$$

where *Operation* represents the three synthesis operations for additive synthesis, ring modulation, and frequency modulation. For each anchor/satellite relationship, we can choose a different operation. Each *Oscillator* also has its own *Waveform*, here a selection of four varying ones, for instance sine, triangle, square, and sawtooth.[13] Sounds designed this way can immediately be played with by using a keyboard controller, as seen in Section 71.4.

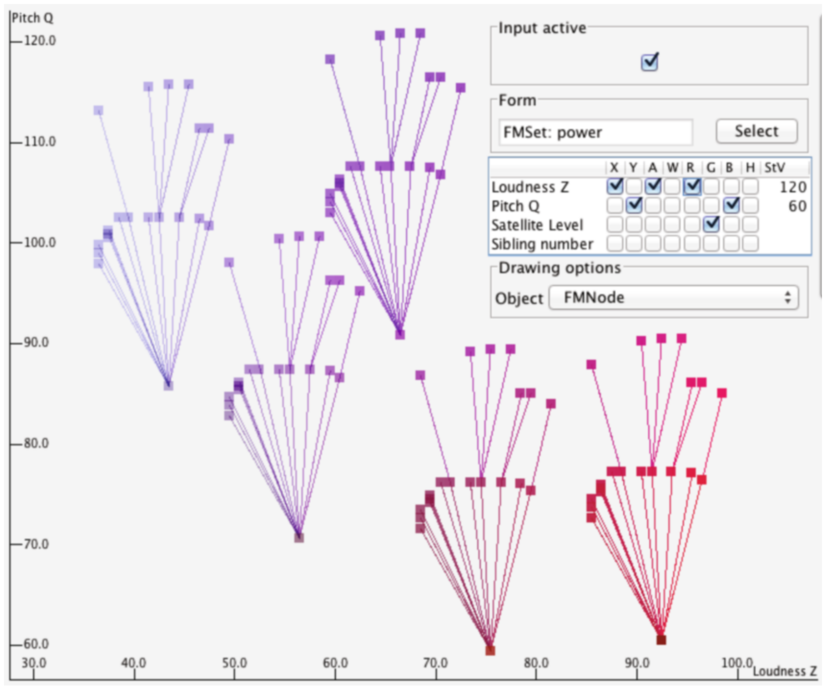
Finally, we can also combine multiple forms into higher-level forms that contain several objects. For instance, a **Limit** of *SoundSpectrum* and *Score* allows us to create compositions containing both constantly sounding pitches and notes with *Onsets* and *Durations*. We simply need to define

$$SpectrumAndScore : .\mathbf{Limit}(Spectrum, Score).$$

Figure 71.16 shows an example of such a composition. This way, any number of synthesis methods and musical formats can be combined to higher-level forms and can be used simultaneously in *BigBang*.

These examples show how much structural variety we can create by just using a small given set of **Simple** forms, and how their visualization can help us understand the structures. All of them can directly

[13] A slightly different *GenericSound* form is described in [1048].



Form: FMSet: power

	X	Y	A	W	R	G	B	H	StV
Loudness Z	✓		✓		✓				120
Pitch Q		✓					✓		60
Satellite Level						✓			
Sibling number									

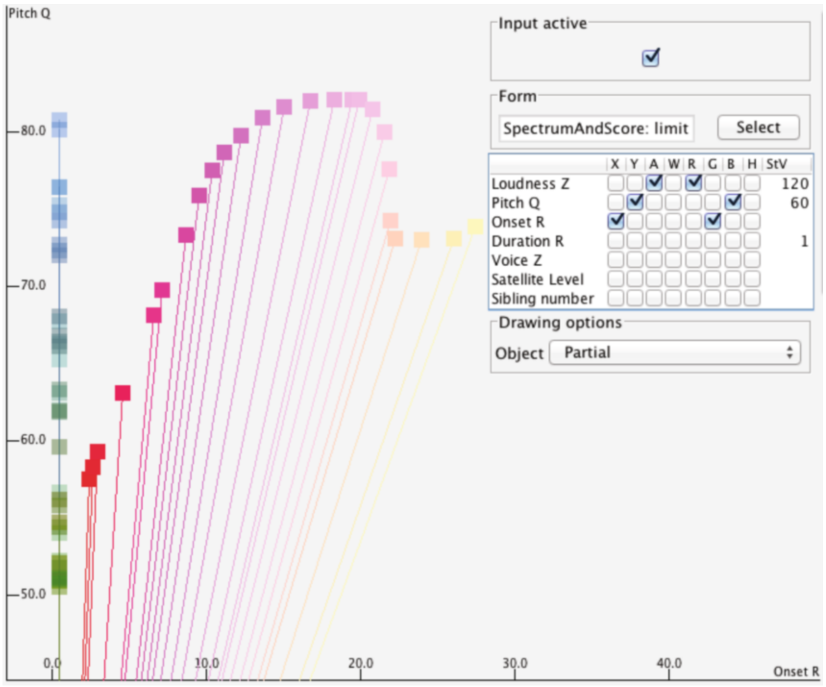
Drawing options

Object: FMNode

Fig. 71.15. An *FMSet* containing five carriers all having the same modulator arrangement, but transposed in *Pitch* and *Loudness*.

be sonified, even while we are building the denotators. Most importantly, such forms can be defined at runtime in *Rubato Composer* and they can immediately be used in *BigBang*. In addition to musical data types, as in the examples here, one can define forms describing any kind of fact. For example, we programmed rubettes that read image files (*ImageFileIn*), translate them into forms, and make them available to transformation in *BigBang*, before being exported again or converted into musical objects by other rubettes.

In the next chapter we will discuss how such objects, once their form is defined, can be created, manipulated, and transformed in *BigBang*. For this, we need to examine how the *BigBang* rubette implements the level of processes.



Input active ✓

Form

SpectrumAndScore: limit

	X	Y	A	W	R	G	B	H	StV
Loudness Z			✓		✓				120
Pitch Q		✓					✓		60
Onset R	✓					✓			
Duration R									1
Voice Z									
Satellite Level									
Sibling number									

Drawing options

Object: Partial

Fig. 71.16. A composition based on a **Limit** of a *SoundSpectrum* (*Pitches* at *Onset* 0) and a *Score* (*Pitches* with *Onsets*).