AUTOMATIC NOTATION OF COMPUTER-GENERATED SCORES FOR INSTRUMENTS, VOICES AND ELECTRO-ACOUSTIC SOUNDS

William A. Burnson, Hans G. Kaper, and Sever Tipei

Computer Music Project, University of Illinois at Urbana-Champaign, Illinois, USA

ABSTRACT

This article proposes a new software system for automatic representation and printing of computer-generated scores using traditional, proportional, and graphic notation. The software system includes a composition module (CMOD) and sound synthesis library (LASS)—both part of DISSCO—and an open source C++ vector-graphics library *Belle, Bonne, Sage* for music engraving. Various challenges to the process of music notation are noted, and a detailed description of the interface between DISSCO and *Belle, Bonne, Sage* is given.

1. BACKGROUND AND REQUIREMENTS

Traditional Western notation reflects the changes Western music went through during the last seven or eight centuries. Among these changes is an increase in the complexity of its notation, culminating in the appearance in the second half of the twentieth century of new symbols and entirely new ways of representing music.

The problem of music notation presents a wide range of challenges for music representation, for various reasons. It renders pitch versus time using symbols that address dynamics, articulations, clefs, harmonics, etc.; absolute proportionality of time and pitch is corrupted by the use of sharps and flats as well as by various symbols for individual durations; the notation of grace notes has more to do with long-established conventions than with logic; etc. These challenges are compounded in some of the more recent schemes, like *proportional notation*, where time, frequency, and sometimes amplitude are seen in graph form; and *graphic notation*, where musical events are presented through a combination of drawings, text, and symbols, similar to the Augenmusik practiced by some Renaissance composers. Electro-acoustic music has added a whole new class of challenges by employing either sounds whose parametric values are incongruent with those present in traditional music (for example, arbitrary frequencies or durations) or "objects" (textures) that are impossible to describe with the tools of the common practice period.

In order to generate the score of a computer-assisted composition for instruments, voices, and/or electro-acoustic sounds, the alphanumeric output resulting from computations needs to be translated into some type of musical notation. Given the fact that even a relatively modest work includes thousands of sounds whose transcription when done by hand takes days, it is easy to see why one would want to avoid this tedious and error-prone activity. Furthermore, a tool for automatic transcription would enable a composer to see the results of his or her labors in a reasonable amount of time.

Automatic transcription cannot be accomplished through the MIDI protocol (as available from some editing applications), because of its severe limitations both with respect to electro-acoustic textures and to contemporary instrumental or vocal extended techniques. A number of applications have addressed the issue, among them GrafPro, developed in 1989 at the University of Illinois Computer Music Project, which dealt successfully with the rhythmic complexities of traditional notation. (GrafPro is no longer in use, since the printing was done through the Score Music Publishing System, which has not kept up with recent developments in technology.) Pure Data (Pd), written by Miller Puckette [6], is capable of rendering graphic images of the electro-acoustic music objects it creates but does not handle traditional notation. Other applications such as Common Music Notation [7], the PWGL-Expressive Notation package [4], or FOMUS [5] have each solved some but not all of the necessary requirements: automatic processing of data; handling of both traditional, proportional, and graphic notation; dealing with instrumental/vocal parts as well as electro-acoustic sounds; and providing high quality, publishable documents.

In this article we describe an automatic music notation interface between the computer-generated output for instruments, voices, and synthesized sounds (DISSCO) and an open source C++ vector-graphics library for music engraving (*Belle, Bonne, Sage*). [1]

2. DISSCO

The software around which this project is built, DISSCO [2], includes a Composition Module (CMOD) and a Library for Additive Sound Synthesis (LASS). Its output consists of music for instruments and voices as well as digitally synthesized sounds. Both categories are defined by a series of familiar parameters: start time, duration, frequency, etc.,

along with more customized ones: transients, harmonics, multiphonics, mutes, spatial location, reverberation, glissandi, articulations and so on. As information about higher-level structures is also available, DISSCO offers the option to output icons to describe collective behavior—such as the "boxes" in the music of Lutoslawski and other aleatory composers, nonstandard symbols, or even drawings and text. Such icons are identified through their type, a required feature of all events. An XML file containing selected information about various structural levels assists the user in monitoring the sequence of events. However, unless interfaced with an engraving tool, monitoring a sequence of events is only the beginning of the daunting task of manually transcribing the output.

3. BELLE, BONNE, SAGE

Belle, Bonne, Sage[1] is a C++ vector-graphics library for music notation and is a recent development in open-source engraving. The library is built on an assumption-free model: all structure is specified by the developer within its C++ abstractions of a score (collection of pages), and all drawing is sent to an abstract painter, which may be a screen, file, or other device. Conveniently, the library implements a reference PDF painter to boot. As PDF has become the most popular document exchange format for viewing and printing, and since the specification contains a well-defined conversion to its ancestor, PostScript, the format becomes a comprehensive solution to the problem of precisely representing music notation in a modern format.

3.1. PDF Output

As PDF technology evolved, three fundamental graphical entities developed independently: lines, paths, and fonts. Music notation requires that these heterogeneous primitives, each implemented with its own highly optimized scanline rendering algorithm, be precisely joined together. For example, a quarter note as typically described by notation software consists of a notehead glyph from a music font and a line for a stem whose length and thickness can be controlled. Even when these two are perfectly superimposed, the difference in rasterization causes graphical anomalies at all but the highest resolutions, and at higher resolutions any lack of precision in the calculation of the stem-notehead join will cause sharp corner artifacts to appear. The calculation is nontrivial, since fonts can use different algorithms to control point size on different platforms. To circumvent these problems, *Belle, Bonne, Sage* draws all shapes, including lines and font glyphs, as complex filled paths by default. The technique ensures that no matter the output device, all graphic primitives are rasterized using the same generic algorithm.

3.2. Assumption-Free Model

Belle, Bonne, Sage acts like a meta-engraver in that, instead of accepting music content for input (for example, draw C4, quarter note, beat one, measure one, in piano), the user develops building blocks that create music content, a result of the assumption-free requirement. However, the library gives the user a choice of built-in assumptions. For example, there is a method for drawing a half note that allows one to specify the object's position, size, stem width and height, angle and proportions of the outer ellipse or the inner rounded rectangular hole, and contains default values for most of these parameters. From this information, the method can return a single outline path containing the union of the stem and notehead. In the event that this method is not sufficient for drawing half notes, the user can simply write another one. Since all objects are graphical rather than structural in nature, there are practically no constraints on what is allowable. By taking this approach, *Belle, Bonne, Sage* gradually increases its repertoire of useful tools for notating music without ever requiring an input whose format is canned.

3.3. Collision Detection

Besides the lower-level primitives such as notes and other fundamental musical glyphs, *Belle, Bonne, Sage* uses sophisticated layout protocols in order to guarantee that there are no overlaps. The library provides several facilities for collision detection at many levels of precision. For example, any two glyphs (or set of glyphs) may be "magnetically" attached at an arbitrary distance by calculating the closest distance, along some line, the two things may coexist without overlapping.

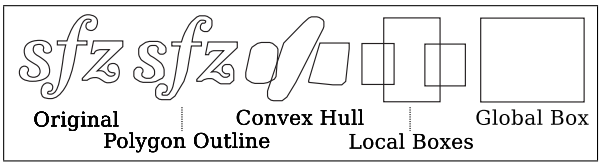

Figure 1. Showing the various representations of a complex glyph used during collision detection. The less precise shapes are used first to rule out as many potential intersections as possible. Progressively finer levels of representation are used up to the precision of the polygon outline.

The collision detection first begins at the level of bounding boxes to rule out trivial cases of non-intersection, and gradually applies higher precision shapes until the error is arbitrarily small. The algorithm guarantees convergence using recursive distance bisection. Higher-level layout is currently being controlled by collision-based anchors which tie (constrain) objects to each other along a connecting string (line). With line-based anchors, objects can adjust dynami-

cally to new siutations following a simple priority scheme.

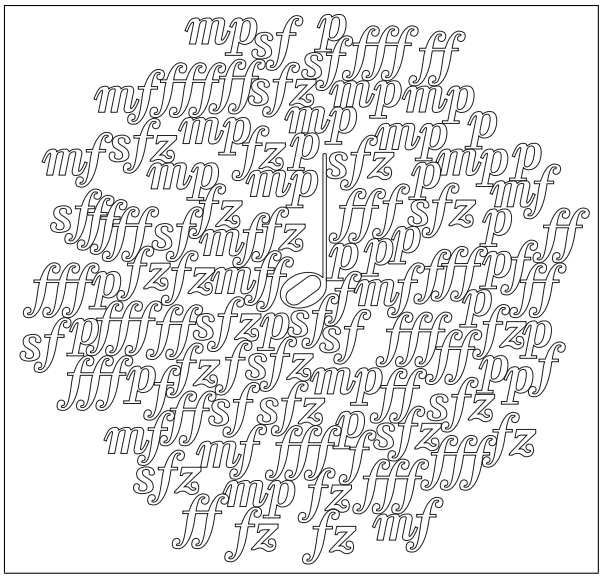

Figure 2. Showing a hundred or so complex glyphs compacted as far as possible—with no overlaps—using precise collision detection. In reality, graphical objects have more space around them: a cinch to adjust once the most-compact-case has been solved for.

4. THE INTERFACE

Since DISSCO produces both note and sound events, *Belle, Bonne, Sage* is ideally suited to generate scores containing any mixture of tape, proportional, and rational (traditional) notation. The resulting scores can be precisely and professionally notated.

An interface formats the alphanumeric output of DISSCO via XML as input for *Belle, Bonne, Sage*. Since the generation of a piece may take an arbitrarily long time, it is useful to save the output in an intermediate format so that one can customize the appearance of the score without having to regenerate the piece. As the transcription and layout algorithms become more sophisticated, the appearance will become more flexible and more options will be given to the user regarding the processing of the intermediate form.

4.1. Intermediate XML Format

The following is an example of the beginning of an XML file generated by DISSCO/CMOD. Elements that contain `<sound>...</sound>` result in displaying an event in the tape part; those that contain `<note>...</note>` will produce notes in the instrumental part.

```
<events>
  <event>
    <name>T/0new</name>
    <type>0</type>
    <start-time-sec>0</start-time-sec>
    <start-time-units>0</start-time-units>
    <duration-sec>409</duration-sec>
    <duration-units>409</duration-units>
    <event>
      <name>H/E0new</name>
      <type>0</type>
      <start-time-sec>0</start-time-sec>
      <start-time-units>0</start-time-units>
      <duration-sec>17</duration-sec>
      <duration-units>17</duration-units>
      <event>
        <name>B/sE0viol00new</name>
        <type>0</type>
        <start-time-sec>0</start-time-sec>
        <start-time-units>0</start-time-units>
        <duration-sec>6.46</duration-sec>
        <duration-units>775.2</duration-units>
        <sound>
          <name>S/adsr.11new</name>
          <type>1</type>
          <global-sound-count>0</global-sound-count>
          <start-time-sec>2.75</start-time-sec>
          <start-time-units>330</start-time-units>
          <duration-sec>1.75</duration-sec>
          <duration-units>210</duration-units>
          <partials>5</partials>
          <sones>252.518</sones>
          <frequency>587.33</frequency>
        </sound>
      </event>
      <event>
        <name>B/n010</name>
        <type>0</type>
        <start-time-sec>2.02726</start-time-sec>
        <start-time-units>8.10902</start-time-units>
        <duration-sec>5</duration-sec>
        <duration-units>20</duration-units>
        <note>
          <start-time-sec>2.02726</start-time-sec>
          <duration-sec>4.20319</duration-sec>
          <pitch>C</pitch>
          <octave>1</octave>
          <loudness>p</loudness>
          <modifier>glissString</modifier>
        </note>
```

4.2. Rhythmic Quantization

Transcribing arbitrary attacks and durations in proportional notation is a simple task, but traditional notation requires the use of select discrete values. DISSCO/CMOD solves this problem by using the lowest common multiple (LCM) of the subdivisions of the beat: if sixteenths (1/4), quintuplets (1/5) and sextuplets (1/6) are present, then LCM = 4 * 5 * 3 = 60 and the beat is divided into 60 units. With the help of sieves [8] only select discrete values are allowed: 15_0 for sixteenths, 12_0 for quintuplets, and 10_0 for sextuplets. Durations are made to conform to the same set of moduli and constrained to end on one of the possible attack points. The same result can be obtained through large Markov matrices covering a few beats at a time. The first procedure is more

deterministic, while the latter makes the selection of the next attack and duration dependent on the previous choice and includes elements of randomness.

Relying on the same type of mod operations, bars, beats, and subdivisions of the beat are identified and durations are split to properly show the way they encompass various beats and bars. **Figure 3** shows a quarter note duration that starts in bar 18 on the third beat (3) after two sixteenths of a quintuplet (24/60 = 2/5), completes the last three sixteenths (36/60 = 3/5) of beat 3 and continues two sixteenths of a quintuplet (24/60 = 2/5) into the fourth beat of the bar.

```
Start:     Bar: 18 units
           Beat: 3 + 24/60 units

Duration:  36/60, 24/60 units
```

Figure 3. Showing the ability of DISSCO/CMOD to correctly spell durations divided between two beats.

4.3. Spacing Coordination

Another dilemma arises when reconciling spacing in combinations of proportional tape notation and traditionally notated instrumental parts. In traditional notation, music is primarily spaced according to the sizes of the graphical objects that take up horizontal space. In proportional spacing, however, a duration assumes a nonnegotiable horizontal size. One way to accommodate both is to ensure that the durations are spaced widely enough that the objects do not collide in the traditional notation, which in some cases can lead to excessively long scores.

A hybrid approach is contemplated so that notes under a certain rhythmic threshold, say sixteenths, may assume extra space as long as they do not push notes of a higher duration off the proportional grid. Or better yet, all objects can be defined to occupy dynamic, static, or minimum width. For example, simple shapes in the electroacoustic part may be easily compressed (perhaps to a minimum legible size) so that they are dependent on the higher priority instrumental mensural notation that will undoubtedly place several constraints the available space due to its much more complex rules of convention.

5. FUTURE WORK

Another well-known problem is that of the correct spelling of accidentals. The FOMUS library [5] contains a sophisticated algorithm for realizing undetermined accidentals in a conventional way by detecting melodic line ascension and descensions. Since the music generated by DISSCO is usually atonal, matters of spelling are generally irrelevant; however, a new CMOD feature is currently under consideration that would allow the user to control how the accidentals are determined.

Although LASS is dedicated to additive synthesis using sine waves, work is presently being done to include other wave types as well as sampled sound objects. When imported samples are used, information about their content or the way in which they were generated is generally not available and a spectral analysis is necessary in order to include them in a printed score. Such a feature could be borrowed from or modeled after Michael Klingbeil's SPEAR [3].

6. REFERENCES

[1] Burnson, W. A., "Introducing Belle, Bonne, Sage." Proc. 2010 Int'l Computer Music Conference, Stony Brook, NY, 2010.

[2] Kaper, H. G. and S. Tipei, "DISSCO: A Unified Approach to Sound Synthesis and Composition." Proc. 2005 Int'l Computer Music Conference, Barcelona, Spain, pp. 375-378.

[3] Klingbeil, M., "Software for Spectral Analysis, Editing, and Synthesis." Proc. 2005 Int'l Computer Music Conference, Barcelona. pp. 107-110.

[4] Kuunskankare, M. and M. Laurson, "Expressive Notation Package." Computer Music Journal, vol. 30 no.4, pp. 67-79, 2006.

[5] Psenicka, D., "Automatic Score Generation with FOMUS." Proc. 2009 Int'l Computer Music Conference, Montreal, Canada, pp. 69-72.

[6] Puckette, M., "Pure Data: Another integrated computer music environment." Proc. Second Intercollege Computer Music Concerts, Tachikawa, Japan, pp. 37-41.

[7] Taube, H., "COMMON MUSIC, A Compositional Language in Common Lisp and CLOS." Proc. 1989 Int'l Computer Music Conference, Columbus Ohio, pp. 316-319.

[8] Tipei, S., "Solving Specific Compositional Problems with MP1." Proc. 1981 Int'l Computer Music Conference, Denton, Texas, 1981, pp. 101-109.