Director Musices is a program that transforms notated scores into musical performances. It implements the performance rules emerging from research projects at the Royal Institute of Technology (KTH). Rules in the program model performance aspects such as phrasing, articulation, and intonation, and they operate on performance variables such as tone, inter-onset duration, amplitude, and pitch. By manipulating rule parameters, the user can act as a metaperformer controlling different features of the performance, leaving the technical execution to the computer. Different interpretations of the same piece can easily be obtained.

Features of Director Musices include MIDI file input and output, rule palettes, graphical display of all performance variables (along with music notation), and user-defined performance rules. The program is implemented in Common Lisp and is available free as a stand-alone application both for Macintosh and Windows platforms. Further information, including music examples, publications, and the software itself, is located online at http://www.speech.kth.se/music/performance/.

Performance Rules

Performance rules previously presented in several articles (e.g., Friberg and Sundberg 1999; Friberg 1991, 1995a, 1995b; Friberg et al. 1991, 1998; Bresin and Friberg 1998; Sundberg 1988, 1993) constitute the core of Director Musices. They are used to modify the nominal values of various performance variables, such as duration and amplitude, as shown in Figure 1.

Most of the rules have a global quantity parameter $k$ (whose default value is unity) that controls the magnitude of all modifications caused by that rule. The effects of various rules may be adjusted by additional parameters. The selection of rules, $k$-values, and rule parameter values can drastically change the performance, and many different but still musically acceptable performances can thus be obtained. An overview of the current rule system is given in Table 1. Previous implementations of a subset of the rules include the Windows program MELODIA (Bresin 1993), and Japer (Bresin and Friberg 1997), a Java program available on the World Wide Web.

One of the goals of our performance research has been to find rules independent of musical style that correspond to basic performance principles used by musicians. The rules can be divided into three categories according to their apparent communicative purpose (Sundberg 1999): (1) grouping rules that mark boundaries between smaller and larger tone groups (e.g., punctuation and phrase arch); (2) differentiation rules that increase differences between categories (e.g., duration contrast); and (3) ensemble rules for the interaction between musicians in an ensemble (e.g., ensemble “swing” and melodic synchronization). Thus the rules are primarily related to basic aspects of performance, such as structural demarcation; yet by rule selection and adjustment of rule parameters, the rules can create performances that differ in emotional quality, for example, “happy” or “sad” (Bresin and Friberg 1998).

Another recent development is the GERM model (Juslin et al. 1999), combining four different performance rule types: generative (described above), emotional, random variations, and associated motion.

Input and Output

Director Musices supports three music formats: (1) scores, a simple text-based custom format; (2) per-
formances, similar to the score format but with added performance variables; and (3) MIDI files. Normally, a new score is entered into an external score editor and then transferred to Director Musices as a MIDI file. The MIDI file reader converts any MIDI file to an internal score object, keeping note durations and assigning a note value to each note for the music notation.

Rules can also be applied to MIDI performances. As each track is assumed to contain only one voice, simultaneous notes in the same track are truncated at a new note onset, thus creating a track suitable for rule application. Key velocities are currently disregarded. Harmonic and phrase analysis, required by some rules as well as other score variables, can be inserted directly in Director Musices.

Score Representation

The representation of the score in Director Musices is straightforward—similar to that of a MIDI file. A score object contains a list of track objects, which in turn contains a list of segments. Each track corresponds to one melodic part, and a segment generally corresponds to one note or one chord. (A chord is any number of simultaneous notes sharing the same performance variables.)

The segment object contains all score and performance variables. The performance variables (except durations) can vary over time by assigning a time-shape object, typically in the form of a breakpoint and an interpolation function. The time shape can be dynamically coupled to a note or phrase chunk. Thus, when the duration of a note is changed, the time shape of the note is scaled accordingly.

Performance variables are expressed in physical measures such as duration in milliseconds and amplitude in decibels. The translation to MIDI variables is made in a synthesizer object—one for each track—making the rule effects independent of the synthesizer used.

Although the program currently generates performances as MIDI files, other output representations such as Csound score files can easily be added. Additionally, the program includes a tool for exporting the performance data to a spreadsheet.

Rule Definition

Most rules require a context. This may consist of a sequence of tones, each with properties such as pitch, inter-onset duration, and harmonic analysis. Some rules operate within a metrical context, while some operate within both vertical (harmonic) and horizontal (melodic) contexts. This context framework was crucial to the choice of score representation and tools for formulating rules.

Instead of creating a complex data structure to describe the music, we chose a simple data structure supplemented by flexible dynamic viewpoints. That is, rules can examine the score at different hierarchical levels and in different chunks. For example, consider a track that represents a vocal instead of an instrumental part. Rather than notes, the track can contain a list of voice segments, each corresponding to a phoneme, such that a note consists of one or several segments. A rule can be applied both at the segment or note level, allowing pronunciation rules to work at the segment level and, at the same time, performance rules at the note level. The performance rules will simply “see” the track as consisting of a sequence of notes, and all accesses to performance variables are the same as for an instrumental track. The different viewpoints are dynamically allocated when a rule is applied, allowing even rule-based selection of chunks.
Table 1. Most of the rules in Director Musices, showing the affected performance variables (sl = sound level, dr = interonset duration, dro = offset to onset duration, va = vibrato amplitude, dc = deviation from equal temperament in cents)

### Marking Pitch Context

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Performance Variables</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-loud</td>
<td>sl</td>
<td>The higher the pitch, the louder</td>
</tr>
<tr>
<td>Melodic-charge</td>
<td>sl dr va</td>
<td>Emphasis on notes remote from current chord</td>
</tr>
<tr>
<td>Harmonic-charge</td>
<td>sl dr</td>
<td>Emphasis on chords remote from current key</td>
</tr>
<tr>
<td>Chromatic-charge</td>
<td>dr sl</td>
<td>Emphasis on notes closer in pitch</td>
</tr>
<tr>
<td>Faster-uphill</td>
<td>dr</td>
<td>Decrease duration for notes in uphill motion</td>
</tr>
<tr>
<td>Leap-tone-duration</td>
<td>dr</td>
<td>Shorten first note of an up-leap and lengthen first note of a down-leap</td>
</tr>
<tr>
<td>Leap-articulation-dro</td>
<td>dro</td>
<td>Micropauses in leaps</td>
</tr>
<tr>
<td>Repetition-articulation-dro</td>
<td>dro</td>
<td>Micropauses in tone repetitions</td>
</tr>
</tbody>
</table>

### Marking Duration and Meter Context

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Performance Variables</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration-contrast</td>
<td>dr sl</td>
<td>The longer the note, the longer and louder; and the shorter the note, the shorter and softer</td>
</tr>
<tr>
<td>Duration-contrast-art</td>
<td>dro</td>
<td>The shorter the note, the longer the micropause</td>
</tr>
<tr>
<td>Double-duration</td>
<td>dr</td>
<td>Decrease duration contrast for two notes with duration relation 2:1</td>
</tr>
<tr>
<td>Social-duration-care</td>
<td>dr</td>
<td>Increase duration for extremely short notes</td>
</tr>
<tr>
<td>Inegales</td>
<td>dr</td>
<td>Long-short patterns of consecutive eighth notes; also called swing eighth notes</td>
</tr>
<tr>
<td>Ensemble-swing</td>
<td>dr</td>
<td>Model different timing and swing ratios in an ensemble proportional to tempo</td>
</tr>
</tbody>
</table>

### Intonation

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Performance Variables</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-sharp</td>
<td>dc</td>
<td>The higher the pitch, the sharper</td>
</tr>
<tr>
<td>Mixed-intonation</td>
<td>dc</td>
<td>Ensemble intonation combining both melodic and harmonic intonation</td>
</tr>
<tr>
<td>Harmonic-intonation</td>
<td>dc</td>
<td>Beat-free intonation of chords relative to root</td>
</tr>
<tr>
<td>Melodic-intonation</td>
<td>dc</td>
<td>Close to Pythagorean tuning, e.g., with sharp leading tones</td>
</tr>
</tbody>
</table>

### Phrasing

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Performance Variables</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punctuation</td>
<td>dr dro</td>
<td>Automatically locates small tone groups and marks them with lengthening of last note and a following micropause</td>
</tr>
<tr>
<td>Phrase-articulation</td>
<td>dro dr</td>
<td>Micropauses after phrase and subphrase boundaries, and lengthening of last note in phrases</td>
</tr>
<tr>
<td>Phrase-arch</td>
<td>dr sl</td>
<td>Each phrase performed with arch-like tempo curve: starting slow, faster in middle, and ritardando toward end; sound level is coupled so that slow tempo corresponds to low sound level</td>
</tr>
<tr>
<td>Final-ritard</td>
<td>dr</td>
<td>Ritardando at end of piece, modeled from stopping runners</td>
</tr>
</tbody>
</table>

### Synchronization

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Performance Variables</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Melodic-sync</td>
<td>dr</td>
<td>Generates new track consisting of all tone onsets in all tracks; at simultaneous onsets, note with maximum melodic charge is selected; all rules applied on this sync track, and resulting durations are transferred back to original tracks</td>
</tr>
<tr>
<td>Bar-sync</td>
<td>dr</td>
<td>Synchronize tracks on each bar line</td>
</tr>
</tbody>
</table>
Other typical viewpoint selections include phrases, measures, and chord progressions.

Rules are written in Common Lisp syntax. Pre-defined functions help rule development, and all standard functions in Common Lisp are available. Some examples of functions and rules are shown below and in Figure 2.

**Rule Top-Level Definition**

Rules are defined by the normal Lisp `defun` special function

```
(defun <rulename> (<k parameter> <additional key parameters>) <body>)
```

This function defines a rule with the main rule parameter `k`. Additional parameters are specified using key parameters.

**Serial Sequencing Functions**

These special functions (Lisp macros) step through the score in chunks as specified by each macro, and are used within the body of a rule definition. The macro

```
(each-note-if <conditions> (then <body>))
```

iterates over each note and track of the score and evaluates `<body>` if all conditions are met. Within the body, access functions are used for note variables. The macro

```
(each-segment-if <conditions> (then <body>))
```

is the same as above, but applied to segments rather than notes. It has the same function as `each-note-if`, provided the track is a mono track. For a voice track, this macro works at a
lower level, where each segment corresponds to a voice segment or a phoneme.

The macro

```
(each-group
  `<group begin condition>
  `<group end condition>
  <body>)
```

first creates a new track consisting of segment group objects (chunks) as specified by the begin

and end conditions, and then evaluates `<body>` for each group.

### Serial Access Functions

Within the body of the sequencing macros, these functions are used for accessing the variables in each chunk. They are also used for defining contexts. A slowly changing variable (a time-shape object) can be applied over an entire chunk.
The macro form given by

\[(\text{this } \text{variable})\]
\[(\text{next } \text{variable})\]
\[(\text{prev } \text{variable})\]

returns the specified variable of the current, next, or previous chunk, while

\[(\text{set-this } \text{variable} \text{value})\]
\[(\text{set-next } \text{variable} \text{value})\]
\[(\text{set-prev } \text{variable} \text{value})\]

assigns the specified value to the variable in the current, next, or previous chunk.

**User Interface**

Figure 3 shows the main windows in the Windows version of Director Musices. The track variables of the score are shown in the second window from the top. Here basic features such as track volume or MIDI program number can be edited.

A performance is defined by selecting rules and rule parameters in a rule palette window. Rule effects are additive, i.e., if a rule is applied twice, the change of the performance variables will be twice as large. Several rule palette windows can be open at the same time, thus allowing easy comparison of different performances.

All performance variables can be shown graphically alongside music notation, as shown in Figure 3. The time axis can either indicate real time or score time.

In addition, the Windows version contains an editable score window where all variables can be edited and displayed along with the music notation. This facilitates the adding of extra information to the score, such as phrase markers.

**Acknowledgments**

Lars Frydén, Johan Sundberg, and Anders Friberg developed most of the rules. Roberto Bresin contributed the duration-contrast articulation rule. Anders Friberg wrote most of the kernel code and the Macintosh version. Vittorio Colombo developed most of the user-interface code for Windows. The project was supported by the Bank of Sweden Tercentenary Foundation.

**References**


