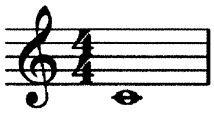
Read starting here up to page 44 for ECE402 homework.

# "Hello World" in MusicXML

Brian Kernighan and Dennis Ritchie popularized the practice of writing a program that prints the words "hello, world" as the first program to write when learning a new programming language. It is the minimal program that tests how to build a program and display its results.

In MusicXML, a song with the lyrics "hello, world" is actually more complicated than we need for a simple MusicXML file. Let us keep things even simpler: a one-measure piece of music that contains a whole note on middle C, based in 4/4 time:



Here it is in MusicXML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML 1.1 Partwise//EN"
    "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="1.1">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

Let's look at each part in turn:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

This is the XML declaration required of all XML documents. We have specified that the characters are written in the Unicode encoding "UTF-8". This is the version of Unicode that has ASCII as a subset. Setting the value of standalone to "no" means that we are defining the document with an external definition in another file.

```
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML 1.1 Partwise//EN"
    "http://www.musicxml.org/dtds/partwise.dtd">
```

This is where we say that we are using MusicXML, specifically a partwise score where measures are contained within parts. We use a PUBLIC declaration including an Internet location for the DTD. The URL in this declaration is just for reference. Most applications that read MusicXML files will want to install s local copy of the MusicXML DTDs on the user's machine. Use the entity resolver in your XML parser to validate against the local copy, rather than reading the DTDs slowly over the network.

```
<score-partwise version="1.1">
```

This is the root document type. The <score-partwise> element is made up of parts, where each part is made up of measures. There is also a <score-timewise> option which is made up of measures, where each measure is made up of parts. The version attribute lets programs distinguish what version of MusicXML is being used more easily. Leave it out if you are writing MusicXML 1.0 files.

```
  <part-list>
    <score-part id="P1">
      <part-name>Part 1</part-name>
    </score-part>
  </part-list>
```

Whether you have a partwise or timewise score, a MusicXML file starts off with a header that lists the different musical parts in the score. The above example is the minimal part-list possible: it contains one score-part, the required id attribute for the score-part, and the required part-name element.

```
  <part id="P1">
```

We are now beginning the first (and only, in this case) part within the document. The id attribute here must refer to an id attribute for a score-part in the header.

```
    <measure number="1">
```

We are starting the first measure in the first part.

```
      <attributes>
```

The attributes element contains key information needed to interpret the notes and musical data that follow in this part.

```
        <divisions>1</divisions>
```

Each note in MusicXML has a duration element. The divisions element provided the unit of measure for the duration element in terms of divisions per quarter note. Since all we have in this file is one whole note, we never have to divide a quarter note, so we set the divisions value to 1.

Musical durations are typically expressed as fractions, such as "quarter" and "eighth" notes. MusicXML durations are fractions, too. Since the denominator rarely needs to change, it is represented separately in the divisions element, so that only the numerator needs to be associated with each individual note. This is similar to the scheme used in MIDI to represent note durations.

```
<key>
  <fifths>0</fifths>
</key>
```

The key element is used to represent a key signature. Here we are in the key of C major, with no flats or sharps, so the fifths element is 0. If we were in the key of D major with 2 sharps, fifths would be set to 2. If we were in the key of F major with 1 flat, fifths would be set to -1. The name "fifths" comes from the representation of a key signature along the circle of fifths. It lets us represent standard key signatures with one element, instead of separate elements for sharps and flats.

```
<time>
  <beats>4</beats>
  <beat-type>4</beat-type>
</time>
```

The time element represents a time signature. Its two component elements, beat and beat-type, are the numerator and denominator of the time signature, respectively.

```
<clef>
  <sign>G</sign>
  <line>2</line>
</clef>
```

MusicXML allows for many different clefs, including many no longer used today. Here, the standard treble clef is represented by a G clef on the second line of the staff (e.g., the second line from the bottom of the staff is a G).

```
</attributes>
<note>
```

We are done with the attributes, and are ready to begin the first note.

```
<pitch>
  <step>C</step>
  <octave>4</octave>
</pitch>
```

The pitch element must have a step and an octave element. Optionally it can have an alter element, if there is a flat or sharp involved. These elements represent sound, so the alter element must always be included if used, even if the alteration is in the key signature. In this case, we have no alteration. The pitch step is C. The octave of 4 indicates the octave the starts with middle C. Thus this note is a middle C.

```
<duration>4</duration>
```

Our divisions value is 1 division per quarter note, so the duration of 4 is the length of 4 quarter notes.

```
<type>whole</type>
```

The <type> element tells us that this is notated as a whole note. You could probably derive this from the duration in this case, but it is much easier to work with both notation and performance applications if the notation and performance data is represented separately.

In any event, the performance and notation data do not always match in practice. For example, if you want to better approximate a swing feel than the equal eighth notes notated in a jazz chart, you might use different duration values while the type remains an eighth note. Bach's music contains examples of shorthand notation where the actual note durations do not match the standard interpretation of the notes on the page, due to his use of a notational shorthand for certain rhythms.

The duration element should reflect the intended duration, not a longer or shorter duration specific to a certain performance. The note element has attack and release attributes that suggest ways to alter a note's start and stop times from the nominal duration indicated directly or indirectly by the score.

```
      </note>
```

We are done with the note.

```
    </measure>
```

We are done with the measure.

```
  </part>
```

We are done with the part.

```
</score-partwise>
```

And we are done with the score.

One limitation of XML's document type definitions is that if you want to limit the number of elements within another element, you generally must also restrict how they are ordered as well. In the attributes, for instance, we want no more than one divisions element; for the note's pitch, we want one and only one step element and octave element. In order to do this, the order in which these elements appear must be constrained as well.

Thus the order in which elements appear in these examples does matter. The DTD definitions should make it clear what ordering is required; we will not spell that out in detail during the tutorial.

# The Structure of MusicXML Files

## Adapting Musical Scores to a Hierarchy

Say we have a piece of music for two or more people to play. It has multiple parts, one per player, and multiple measures. XML represents data in a hierarchy, but musical scores are more like a lattice. How do we reconcile this? Should the horizontal organization of musical parts be primary, or should the vertical organization of musical measures?

The answer is different for every music application. David Huron, a music cognition specialist and the inventor of Humdrum, advised us to make sure we could represent music both ways, and be able to switch between them easily.

This is why MusicXML has two different top-level DTDs, each with its own root element. If you use the partwise DTD, the root element is <score-partwise>. The musical part is primary, and measures are contained within each part. If you use the timewise DTD, the root element is <score-timewise>. The measure is primary, and musical parts are contained within each measure.

Having two different structures does not work well if there is no automatic way to switch between them. MusicXML provides two XSLT stylesheets to convert back and forth between the two document types. The parttime.xsl stylesheet converts from <score-partwise> to <score-timewise>, while the timepart.xsl stylesheet converts from <score-timewise> to <score-partwise>.

An application reading MusicXML can choose which format is primary, and check for that document type. If it is your root element, just proceed. If not, check to see if it is the other MusicXML root element. If so, apply the appropriate XSLT stylesheet to create a new MusicXML document in your preferred format, and then proceed. If it is neither of the two top-level document types, you do not have a MusicXML score, and can return an appropriate error message.

When your application writes to MusicXML, simply write to whichever format best meets your needs. Let the program reading the MusicXML convert it if necessary. If you have a two-dimensional organization in your program so that either format is truly equally easy to write, consider using the score-partwise format. Most of today's MusicXML software uses this format, so if all else is equal, conversion times should be lower overall.

## Top-Level Document Elements

The score.dtd file defines the basic structure of a MusicXML file. The primary definition of the file is contained in these lines:

```
<![ %partwise; [
<!ELEMENT score-partwise (%score-header;, part+)>
<!ELEMENT part (measure+)>
<!ELEMENT measure (%music-data;)>
]]>
<![ %timewise; [
<!ELEMENT score-timewise (%score-header;, measure+)>
<!ELEMENT measure (part+)>
<!ELEMENT part (%music-data;)>
]]>
```

The %partwise; and %timewise; entities are set in the top-level DTDs partwise.dtd and timewise.dtd. The <![ lines indicate a conditional clause like the #ifdef statement in C. So if partwise.dtd is used, the <score-partwise> element is defined, while if timewise.dtd is used, the <score-timewise> element is defined.

You can see that the only difference between the two formats is the way that the part and measure elements are arranged. A score-partwise document contains one or more part elements, and each part element contains one or more measure elements. The score-timewise document reverses this ordering.

In either case, the lower-level elements are filled with a music-data entity. This contains the actual music in the score, and is defined as:

```
<!ENTITY % music-data
        "(note | backup | forward | direction | attributes |
          harmony | figured-bass | print | sound | barline |
          grouping | link | bookmark)*">
```

In addition, each MusicXML file contains a %score-header; entity, defined as:

```
<!ENTITY % score-header
        "(work?, movement-number?, movement-title?,
          identification?, defaults?, credit*, part-list)">
```

We will now look at the score-header entity in more detail. If the example in the preceding "Hello World" section gave you enough information, you may want to skip ahead to the next section that starts describing music-data.

# The Score Header Entity

The score header contains some basic metadata about a musical score, such as the title and composer. It also contains the part-list, which lists all the parts or instruments in a musical score.

As an example, take our MusicXML encoding of "Mut," the 22nd song from Franz Schubert's song cycle *Winterreise*. Here is a sample score header for that work:

```
<work>
  <work-number>D. 911</work-number>
  <work-title>Winterreise</work-title>
</work>
<movement-number>22</movement-number>
<movement-title>Mut</movement-title>
<identification>
  <creator type="composer">Franz Schubert</creator>
  <creator type="poet">Wilhelm Müller</creator>
  <rights>Copyright © 2001 Recordare LLC</rights>
  <encoding>
    <encoding-date>2002-02-16</encoding-date>
    <encoder>Michael Good</encoder>
    <software>Finale 2002 for Windows</software>
    <encoding-description>MusicXML 1.0 example</encoding-description>
  </encoding>
  <source>Based on Breitkopf &amp;amp; Härtel edition of 1895</source>
</identification>
<part-list>
  <score-part id="P1">
    <part-name>Singstimme.</part-name>
  </score-part>
  <score-part id="P2">
```

```
      <part-name>Pianoforte.</part-name>
    </score-part>
  </part-list>
```

You see that this score-header has all five of the possible top-level elements in the score-header entity: the work, movement-number, movement-title, identification, and part-list. Only the part-list is required, all other elements are optional.

Let's look at each part in turn:

```
  <work>
    <work-number>D. 911</work-number>
    <work-title>Winterreise</work-title>
  </work>
```

In MusicXML, individual movements are usually represented as separate files. The work element is used to identify the larger work of which this movement is a part. Schubert's works are more commonly referred to via D. numbers than opus numbers, so that is what we use in the work-number element; the work-title is the name of the larger work.

If you have all the movements in a work represented, you can use the opus element to link to the MusicXML opus file that in turn contains links to all the movements in the work.

```
  <movement-number>22</movement-number>
```

*Winterreise* is a cycle of 24 songs. We use the movement-number to identify that "Mut" is the 22nd song in the cycle - it is not restricted to use for movements in a symphony.

```
  <movement-title>Mut</movement-title>
```

Similarly, we use the movement-title element for the title of the individual song. If you have a single song that is not part of a collection, you will usually put the title of the song in the movement-title element, and not use either the work or movement-number elements.

```
  <identification>
    <creator type="composer">Franz Schubert</creator>
    <creator type="poet">Wilhelm Müller</creator>
```

The identification element is defined in the identity.dtd file. It contains basic metadata elements based on the Dublin Core. In this song, as many others, there are two creators: in this case, the composer and the poet. Therefore, we use two creator elements, and distinguish their roles with the type attribute. For an instrumental work with just one composer, there is no need to use the type attribute.

```
    <rights>Copyright © 2001 Recordare LLC</rights>
```

The rights element contains the copyright notice. You may have multiple rights elements if multiple copyrights are involved, say for the words and the music. As with the creator element, these can have type attributes to indicate what type of copyright is involved. In this example, both the words and music to Mut are in the public domain, but we are copyrighting our electronic edition of the work.

```
    <encoding>
      <encoding-date>2002-02-16</encoding-date>
      <encoder>Michael Good</encoder>
      <software>Finale 2002 for Windows</software>
      <encoding-description>MusicXML 1.0 example</encoding-description>
    </encoding>
```

The encoding element contains information about how the MusicXML file was created. Here we are using all four of the available sub-elements to describe the encoding. You can have multiple instances of these elements, and they can appear in any order.

```
    <source>Based on Breitkopf &amp;amp; Härtel edition of 1895</source>
  </identification>
```

The source element is useful for music that is encoded from a paper published score or manuscript. Different editions of music will contain different musical information. In our case, we used the Dover reprint of the Breitkopf & Härtel edition of *Winterreise* as our starting point, correcting some errors in that published score.

The identification element also may contain a miscellaneous element. This in turn contains miscellaneous-field elements, each with a name attribute. This can be helpful if your software contains some identification information not present in the MusicXML DTD that you want to preserve when saving and reading from MusicXML.

```
<part-list>
  <score-part id="P1">
    <part-name>Singstimme.</part-name>
  </score-part>
  <score-part id="P2">
    <part-name>Pianoforte.</part-name>
  </score-part>
</part-list>
```

The part-list is the one part of the score header that is required in all MusicXML scores. It is made up of a series of score-part elements, each with a required id attribute and part-name element. By convention, our software simply numbers the parts as "P1", "P2", etc. to create the id attributes. You may use whatever technique you like as long as it produces unique names for each score-part.

In addition to the part-name, there are many optional elements that can be included in a score-part:

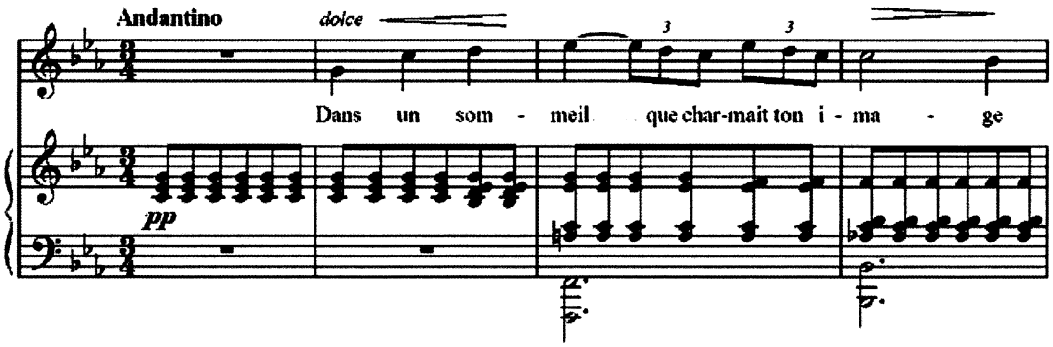
- An identification element, helpful if individual parts come from different sources.
- A part-abbreviation element. Often, you will use the part-name for the name used at the start of the score, and the part-abbreviation for the abbreviated name used in succeeding systems.
- A group element, used when different parts can be used for different purposes. In MuseData, for instance, there will often be different parts used for a printed score, a printed part, a MIDI sound file, or for data analysis.
- One or more score-instrument elements, used to describe multiple instruments within a score-part. This element serves as a reference point for MIDI instrument changes.
- A midi-device element for identifying the MIDI device or port that is being used in a multi-port configuration. Multiple devices let you get beyond MIDI's 16-channel barrier.
- One or more midi-instrument elements, specifying the initial MIDI setup for each score-instrument within a part.

# The MIDI-Compatible Part of MusicXML

MusicXML music data contains two main types of elements. One set of elements is used primarily to represent how a piece of music should sound. These are the elements that are used when creating a MIDI file from MusicXML. The other set is used primarily to represent how a piece of music should look. These elements are used when creating a Finale file from MusicXML.

We encourage programs writing MusicXML to write as much accurate data as they can. The only elements that are required, though, are the sounding elements that relate directly to writing a MIDI file from MusicXML. This is where we will start in introducing the musical elements of a MusicXML file.

As an example, we will use the first four bars of "Après un rêve" by Gabriel Fauré:




## Attributes

The attributes element contains information about time signatures, key signatures, transpositions, clefs, and other musical data that is usually specified at the beginning of a piece or at the start of a measure. We discuss the MIDI-compatible elements here; the rest are discussed in the following sections.

In this example, our Finale translator produces the following MIDI-compatible attributes:

```
<attributes>
  <divisions>12</divisions>
    <key>
      <fifths>-3</fifths>
      <mode>minor</mode>
    </key>
  <time>
    <beats>3</beats>
    <beat-type>4</beat-type>
  </time>
</attributes>
```

### Divisions

Musical durations are commonly referred to as fractions: whole notes, half notes, quarter notes, and the like. While each musical note could have a fraction associated with it, MusicXML instead

follows MIDI by specifying the number of divisions per quarter note at the start of a musical part, and then specifying note durations in terms of these divisions.

MusicXML allows divisions to change in the middle of a part, but most software will probably find it easiest to compute one divisions value per part and put that at the beginning of the first measure. The divisions value of 12 in this example allows for both triplet eighth notes (duration of 4) and regular sixteenth notes (duration of 3).

### Key

Standard key signatures are represented very much like MIDI key signatures. The fifths element specifies the number of flats or sharps in the key signature - negative for flats, positive for sharps. The fifths name indicates that this value represents the key signature's position on the circle of fifths. MusicXML uses the mode element to indicate major or minor key signatures.

### Time

Standard time signatures are represented more simply in MusicXML than in MIDI. The beats element represents the time signature numerator, and the beat-type element represents the time signature denominator (vs. a log denominator in MIDI).

### Transpose

If you are writing a part for a transposing instrument, the transposition must be specified in MusicXML in order for the sound output to be correct. The transpose element represents what must be added to the written pitch to get the correct sounding pitch.

The chromatic element, representing the number of chromatic steps to add to the written pitch, is the one required element. The diatonic, octave-change, and double elements are elements.

Say we have a part written for a trumpet in B-flat. A written "C" on this part will sound as a B-flat on a piano. This transposition is one diatonic step down (C to B) and two chromatic half steps down (C to B to B-flat). In MusicXML it would be represented as:

```
<transpose>
  <diatonic>-1</diatonic>
  <chromatic>-2</chromatic>
</transpose>
```

The diatonic element is not needed for correct MIDI output, but it helps get transposition notation correct and programs are encouraged to use it wherever possible.

The octave-change element is used when transpositions exceed an octave in either direction. The double element is used when the part should be doubled an octave lower, as when a single part is used for both cello and string bass.

## Pitch

Pitch, duration, ties, and lyrics are all represented within the MusicXML note element. For example, the E-flat that starts bar 3 in the voice part has the following MIDI-compatible elements:

```
<note>
```

```
    <pitch>
      <step>E</step>
      <alter>-1</alter>
      <octave>5</octave>
    </pitch>
    <duration>12</duration>
    <tie type="start"/>
    <lyric>
      <syllabic>end</syllabic>
      <text>meil</text>
    </lyric>
  </note>
```

In MIDI, a pitch is represented by a single number. MusicXML divides pitch up into three parts: the step element (A, B, C, D, E, F, or G), an optional alter element (-1 for flat, 1 for sharp), and an octave element (4 for the octave starting with middle C).

The pitch represents the sound, not what is notated, so an alter element must be included even if represents a flat or sharp that is part of the key signature. This is why the E-flat contains an alter element, though there is no accidental on the note.

Alter values of -2 and 2 can be used for double-flat and double-sharp. Decimal values can be used for microtones (e.g., 0.5 for a quarter-tone sharp), but not all programs may convert this into MIDI pitch-bend data.

For rests, a rest element is used instead of the pitch element. The whole rest in 3/4 that begins the voice part is represented as:

```
  <note>
    <rest/>
    <duration>36</duration>
  </note>
```

## Duration

The duration element is an integer that represents a note's duration in terms of divisions per quarter note. Since our example has 12 divisions per quarter note, a quarter note has a duration of 12. The eighth-note triplets have a duration of 4, while the eighth notes have a duration of 6.

## Tied Notes

The sounding part of a tied note is indicated by the tie element. The tie element has a type of start for the starting note of a tie, and a type of stop for the ending note in a tie. A note element can have two tie elements. If a note is tied to the notes both before and after it, place the tie to the previous note, <tie type="stop">, before the <tie type="start"> to the next note.

## Chords

The duration elements in MusicXML move a musical counter. To play chords, we need to indicate that a note should start at the same time as the previous note, rather than following the previous note. To do this in MusicXML, add a chord element to the note.

In our example, the piano part does not have rhythms more complex than eighth notes, so our converter sets the divisions value to 2. With 2 divisions per quarter note, the sound portion of the first chord in the piano part is represented as:

```
<note>
  <pitch>
    <step>C</step>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
</note>
<note>
  <chord/>
  <pitch>
    <step>E</step>
    <alter>-1</alter>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
</note>
<note>
  <chord/>
  <pitch>
    <step>G</step>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
</note>
```

Each note in the chord following the first one includes a chord element before the pitch element.

If you find that you have notes in a chord with different durations, you are probably better representing this as multi-part music rather than a chord. If you must have notes with different durations in the chord, the longest note must be the first note in the chord.

## Lyrics

While lyrics are not yet used in sound generation, they are included in Standard MIDI files, so we will discuss them here with the other MIDI-compatible features of MusicXML.

Lyrics in MusicXML use an optional syllabic element to indicate how a syllable fits into a word, rather than having conventions based on hyphens and spaces as some other formats do. The values for syllabic can be "single", "begin", "end", or "middle". We saw earlier that the E-flat starting the third measure had a syllabic value of "end", since "meil" was the end of a two-syllable word. The "ma" syllable in "image" has a syllabic value of "middle". In the second measure, the notes are:

```
<note>
  <pitch>
    <step>G</step>
    <octave>4</octave>
  </pitch>
  <duration>12</duration>
  <lyric>
    <syllabic>single</syllabic>
    <text>Dans</text>
  </lyric>
</note>
<note>
  <pitch>
    <step>C</step>
    <octave>5</octave>
```

```
    </pitch>
    <duration>12</duration>
    <lyric>
      <syllabic>single</syllabic>
      <text>un</text>
    </lyric>
  </note>
  <note>
    <pitch>
      <step>D</step>
      <octave>5</octave>
    </pitch>
    <duration>12</duration>
    <lyric>
      <syllabic>begin</syllabic>
      <text>som</text>
    </lyric>
  </note>
```

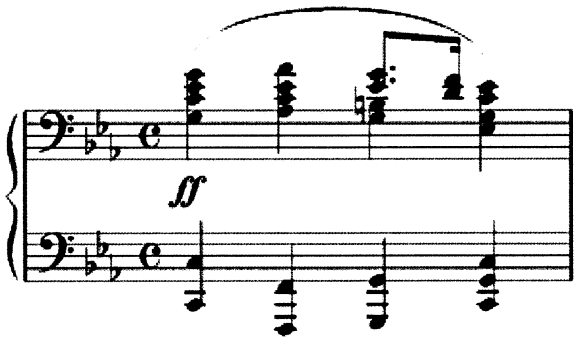
The actual text of the lyric is specified in the text element. A note may have multiple syllables, in which case the multiple syllabic/text element pairs should be separated by an elision element. Word extensions may be indicated by using the extend element.

Multiple verses are indicating using multiple lyric elements. The number and name attributes can be used to distinguish them: <lyric number="1"> for the first verse, <lyric number="2"> for the second.

MusicXML has end-line and end-paragraph elements to support Standard MIDI File Lyric meta-events specified in RP-017. These are used for karaoke and similar applications. Elements for humming and laughing may also be included, though they do not have MIDI equivalents. These lyric elements have not yet been implemented in MusicXML software.

## Multi-Part Music

While monophonic instruments like trumpet, flute, and voice move along one note at a time, instruments like the piano can have many musical lines at once. Take this simple example from the first bar of Frederic Chopin's Prelude, Op. 28, No. 20:




Within the piano part, there are two musical lines for the left hand and right hand, represented in the two staves. On the third beat of the bar, the right hand divides into two lines as well.

We mentioned earlier that the duration element in a note moves the MusicXML musical counter, and that a chord element keeps this counter from moving further. In order to represent parallel

musical parts, we need to be able to move the musical counter forwards and backwards independently of notes. This is what the forward and backup elements let us do.

Let's say we have 4 divisions per quarter in this example. We could approach the divided parts in the right hand in two ways. Finale, for instance, can represent multiple parts using either the layer feature or the voice 1/voice 2 feature. When using layers, each independent part generally covers a complete measure. Say that the G and B-natural on beat 3 are in layer 2, with all the other notes in layer 1. After completing the last chord in layer 1, we would use the following to add the layer 2 notes:

```
<backup>
  <duration>16</duration>
</backup>
<forward>
  <duration>8</duration>
</forward>
<note>
  <pitch>
    <step>G</step>
    <octave>3</octave>
  </pitch>
  <duration>4</duration>
</note>
<note>
  <chord/>
  <pitch>
    <step>B</step>
    <octave>3</octave>
  </pitch>
  <duration>4</duration>
</note>
<forward>
  <duration>4</duration>
</forward>
```

The first backup element moves the counter back to the beginning of the measure. The forward element that follows serves as an invisible half rest. The two note elements provide the quarter note chord. The last forward element serves as an invisible quarter note rest to move the music counter up to the end of the measure.

On the other hand, we could use the voice 1/voice 2 feature on the third beat to indicate a temporary split in the parts. In this case, we would have something like:

```
<note>
  <pitch>
    <step>G</step>
    <octave>3</octave>
  </pitch>
  <duration>4</duration>
</note>
<note>
  <chord/>
  <pitch>
    <step>B</step>
    <octave>3</octave>
  </pitch>
  <duration>4</duration>
</note>
<backup>
```

```
    <duration>4</duration>
  </backup>
  <note>
    <pitch>
      <step>E</step>
      <alter>-1</alter>
      <octave>4</octave>
    </pitch>
    <duration>3</duration>
  </note>
  <note>
    <chord/>
    <pitch>
      <step>G</step>
      <octave>4</octave>
    </pitch>
    <duration>3</duration>
  </note>
  <note>
    <pitch>
      <step>D</step>
      <octave>4</octave>
    </pitch>
    <duration>1</duration>
  </note>
  <note>
    <chord/>
    <pitch>
      <step>F</step>
      <octave>4</octave>
    </pitch>
    <duration>1</duration>
  </note>
```

Here, all that is needed is a backup command to go backup over the quarter note so that the dotted eighth and sixteenth are positioned properly.

MusicXML can handle either type of multi-part writing. In Finale, most users find it easier to use layers rather than voice 1/voice 2, so our Finale-based examples will use that idiom more often. But the choice of what to use is up to you, based on what fits your software best.

In both cases, we would then use a <backup> element with a duration of 16 to move from the end of the first staff to the beginning of the second staff, where we can continue with the left-hand line. MusicXML offers more features for multi-part and multi-staff writing that will be described in later sections, but the elements listed here are all that is needed to multi-part sound output.

# Repeats

Repeats and endings are represented by the <repeat> and <ending> elements with a <barline>, as defined in the barline.dtd file.

In regular measures, there is no need to include the <barline> element. It is only need to represent repeats, endings, and graphical styles such as double barlines.

A forward repeat mark is represented by a left barline at the beginning of the measure (following the attributes element, if there is one):

```
<barline location="left">
```

```
    <bar-style>heavy-light</bar-style>
    <repeat direction="forward"/>
  </barline>
```

The repeat element is what is used for sound generation; the bar-style element only indicates graphic appearance.

Similarly, a backward repeat mark is represented by a right barline at the end of the measure:

```
  <barline location="right">
    <bar-style>light-heavy</bar-style>
    <repeat direction="backward"/>
  </barline>
```

While repeats can have forward or backward direction, endings can have three different type attributes: start, stop, and discontinue. The start value is used at the beginning of an ending, at the beginning of a measure. A typical first ending starts like this:

```
  <barline location="left">
    <ending type="start" number="1"/>
  </barline>
```

The stop value is used when the end of the ending is marked with a downward hook, as is typical for a first ending. It is usually used together with a backward repeat at the end of a measure:

```
  <barline location="right">
    <bar-style>light-heavy</bar-style>
    <ending type="stop" number="1"/>
    <repeat direction="backward"/>
  </barline>
```

The discontinue value is typically used for the last ending in a set, where there is no downward hook to mark the end of an ending:

```
  <barline location="right">
    <ending type="discontinue" number="2"/>
  </barline>
```

# Sound Suggestions

Musical scores abound with ambiguous notations for dynamics, tempo, and other musical elements. To automatically generate a MIDI or other sound file, some value must be used for dynamics or tempo. MusicXML defaults to a MIDI velocity of 90 (roughly a forte); no default tempo is specified.

Sound suggestion elements and attributes guide the creation of a sound file. Most of the sound suggestions are found in the sound element, defined in the direction.dtd file. Tempo is specified in quarter notes per minute. Dynamics are specified as a percentage of the standard MusicXML forte velocity. For example, the following sound element specifies a tempo of quarter note = 88 with a MIDI velocity of 64:

```
  <sound tempo="88" dynamics="71"/>
```

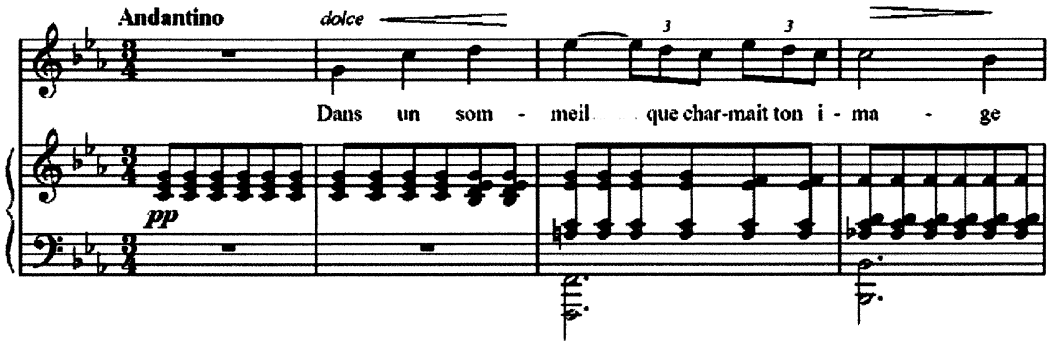
The other attributes for the sound element are described in the direction.dtd file. Sound suggestions are also available for grace notes (the steal-time-previous, steal-time-following, and make-time attributes, defined in the note.dtd file) and for ornaments (the trill-sound entity for trills and other ornaments, defined in the common.dtd file).

# Notation Basics in MusicXML

MIDI represents musical performance information, but leaves out a great deal of information about music notation. MusicXML represents this information, making it much more useful than MIDI for interchange between notation programs. In this section we describe the main elements used to represent music notation that go far beyond what is represented in MIDI files.

## How Music Looks vs. How Music Sounds

Let us look again at the example we used in the previous section - the first four bars of "Après un rêve" by Gabriel Fauré:




Clearly our discussion of the MIDI-compatible portion of MusicXML left out many things represented in this music. Where are the tempo and dynamic markings: the Andantino, pp, dolce, crescendo and diminuendo wedges? Where are stem directions stored? The downstem on the initial G in the voice part is not what many programs would default to. How is the beaming represented, so that all the eighth notes are beamed together in the piano part, but separated into triplets in the voice part? How are the piano chords split between staves? How are accidentals indicated, including courtesy accidentals like the A-flat in the fourth bar?

A fundamental part of MusicXML is the distinction between elements that primarily represent the sound of the music versus those that represent its appearance. We discussed the sound elements in the previous section, and they are of great use to applications dealing with MIDI or other sound files. Now we discuss the elements for musical appearance, which are of great use to music notation applications.

Here is what the beginning of the voice part looks like for "Après un rêve," up to the end of the first measure:

```
<part id="P1">
  <measure number="1">
    <attributes>
      <divisions>12</divisions>
      <key>
        <fifths>-3</fifths>
        <mode>minor</mode>
      </key>
      <time>
        <beats>3</beats>
        <beat-type>4</beat-type>
      </time>
```

```
      <clef>
        <sign>G</sign>
        <line>2</line>
      </clef>
    </attributes>
    <direction placement="above">
      <direction-type>
        <words font-weight="bold" relative-x="-40">Andantino</words>
      </direction-type>
      <sound tempo="80"/>
    </direction>
    <note>
      <rest/>
      <duration>36</duration>
      <voice>1</voice>
    </note>
  </measure>
```

And here is what the beginning of the piano part looks like for "Après un rêve," up to the first chord in the piano part. We will discuss the appearance elements used in these two examples in the rest of this section.

```
<part id="P2">
  <measure number="1">
    <attributes>
      <divisions>2</divisions>
      <key>
        <fifths>-3</fifths>
        <mode>minor</mode>
      </key>
      <time>
        <beats>3</beats>
        <beat-type>4</beat-type>
      </time>
      <staves>2</staves>
      <clef number="1">
        <sign>G</sign>
        <line>2</line>
      </clef>
      <clef number="2">
        <sign>F</sign>
        <line>4</line>
      </clef>
    </attributes>
    <note>
      <pitch>
        <step>C</step>
        <octave>4</octave>
      </pitch>
      <duration>1</duration>
      <voice>1</voice>
      <type>eighth</type>
      <stem>up</stem>
      <staff>1</staff>
      <beam number="1">begin</beam>
      <notations>
        <dynamics placement="below" relative-y="-25" relative-x="-11">
          <pp/>
        </dynamics>
      </notations>
```

```
</note>
<note>
  <chord/>
  <pitch>
    <step>E</step>
    <alter>-1</alter>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
  <voice>1</voice>
  <type>eighth</type>
  <stem>up</stem>
  <staff>1</staff>
</note>
<note>
  <chord/>
  <pitch>
    <step>G</step>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
  <voice>1</voice>
  <type>eighth</type>
  <stem>up</stem>
  <staff>1</staff>
</note>
```

# Attributes

## Staves

The staves element indicates the number of staves in a musical part, which in this case is 2 staves for the piano part. The staves element is optional. If it is not present, as is the case in the voice part, there is 1 staff for the part.

## Clef

The clef element is used to indicate the clef for the staff. By specifying the clef's sign and its line, MusicXML handles both the common treble and bass clefs along with tenor, alto, percussion, tab, and older clefs. The treble clef definition indicates that the second line from the bottom of the staff is a G; the bass clef definition indicates that the fourth line from the bottom of the staff is an F. The number attribute indicates the staff number if the part has more than one staff.

The clef element may also contain a clef-octave-change element after the line element. This is used for clefs that are written either an octave higher or lower than sounding pitch. For example, the tenor line in choral music is usually written in treble clef, an octave higher than the notes actually sound. The clef for this part would be represented as:

```
<clef>
  <sign>G</sign>
  <line>2</line>
  <clef-octave-change>-1</clef-octave-change>
</clef>
```

While attributes usually appear at the start of a measure, they can appear anywhere within the measure. Mid-measure clef changes are the main use for this feature.

### Time

To represent common and cut time signatures, use the symbol attribute of the time element. For common time, use:

```
<time symbol="common">
  <beats>4</beats>
  <beat-type>4</beat-type>
</time>
```

For cut time, use:

```
<time symbol="cut">
  <beats>2</beats>
  <beat-type>2</beat-type>
</time>
```

Without the symbol attribute, these time signatures would appear as 4/4 and 2/2, respectively.

### Directive

A tempo indication at the start of the piece may be indicated either by a directive within the attributes element, or by a direction element discussed below. The "Andantino." starting the piano part could also be represented as:

```
<directive>Andantino.</directive>
```

within the attributes element. The choice is up to the software. Some file formats work more easily with one or the other representation.

## Musical Directions

Musical directions are used for the expression marks in a musical score that are not clearly tied to a particular note. The beginning of the voice part in measure 2, for instance, looks like this:

```
<direction placement="above">
  <direction-type>
    <words font-style="italic" relative-x="-12">dolce</words>
  </direction-type>
</direction>
<note>
  <pitch>
    <step>G</step>
    <octave>4</octave>
  </pitch>
  <duration>12</duration>
  <voice>1</voice>
  <type>quarter</type>
  <stem>down</stem>
  <lyric number="1">
    <syllabic>single</syllabic>
    <text>Dans</text>
  </lyric>
</note>
<direction placement="above">
  <direction-type>
    <wedge type="crescendo" spread="0" relative-y="5"/>
  </direction-type>
  <offset>-4</offset>
</direction>
```

This indicates that the "dolce" mark starts a little before the first note, while the crescendo wedge starts two-thirds of the way between the first and second notes in the measure. The placement attribute is used to indicate whether the directions go above or below the staff. The relative-x and relative-y attributes use units of tenths of interline space. The offset element measures horizontal direction in terms of divisions, just like the duration element.

If two directions go together, they can be linked by having multiple direction-type elements within a single direction. A MusicXML direction-type can contain many different elements, including words, dynamics, wedge, segno, coda, rehearsal, dashes, pedal, metronome, and octave-shift (for 8va and related marks). Elements that continue over time have a type attribute to indicate the start and end points. For a wedge, the type may be crescendo, diminuendo, or stop. For octave-shift, the choice is up, down, or stop. The shift indicates whether the note appears up or down from the sounding pitch, so the start of an 8va has a type of down. For dashes and pedal, the choice is start or stop.

# Note Appearance

## Symbolic Note Types

Given the duration of a note and the divisions attribute, a program can usually infer the symbolic note type (e.g. quarter note, dotted-eighth note). However, it is much easier for notation programs if this is represented explicitly, rather than making the program infer the correct symbolic value. In some cases, the intended note duration does not match what is written, be it some of Bach's dotted notations, notes inégales, or jazz swing rhythms.

The type element is used to indicate the symbolic note type, such as quarter, eighth, or 16th. MusicXML symbolic note types range from 256th notes to long notes: 256th, 128th, 64th, 32nd, 16th, eighth, quarter, half, whole, breve, and long. The type element may be followed by one or more empty dot elements to indicate dotted notes.

## Tuplets

The time-modification element is used to make it easier for applications to handle tuplets properly. For a normal triplet, this would look like:

```
<time-modification>
  <actual-notes>3</actual-notes>
  <normal-notes>2</normal-notes>
</time-modification>
```

This indicates that three notes are placed in the time usually allotted for two notes.

There is an optional normal-type element that is used when the type of the note does not match the type of the normal-notes in the triplet. Say you have an eighth note triplet, but instead of three eighth notes, you have a quarter note and eighth note instead. Without a normal-type element, software that reads the quarter note in the tuplet will likely assume that this is starting a quarter-note triplet, not an eighth note triplet. In this case, the symbolic type and tuplet would be encoded as:

```
<type>quarter</type>
<time-modification>
  <actual-notes>3</actual-notes>
  <normal-notes>2</normal-notes>
  <normal-type>eighth</normal-type>
```

```
</time-modification>
```

## Stems

Stem direction is represented with the stem element, whose value can be up, down, none, or double.

## Beams

Beams are represented by beam elements. Their value can be begin, continue, end, forward hook, and backward hook. Each element has a beam-level attribute which ranges from 1 to 6 for eighth-note to 256th-note beams.

## Accidentals

The accidental element represents actual notated accidentals. Values can be sharp, flat, natural, double-sharp, sharp-sharp, flat-flat, natural-sharp, natural-flat, quarter-flat, quarter-sharp, three-quarters-flat, and three-quarters-sharp. An accidental element has optional courtesy and editorial attributes to indicate courtesy and editorial accidentals.

## Notations

Many additional elements can be associated with a note. In MusicXML, these are collected under the notations object. Tied notes, slurs, tuplets, fermatas, and arpeggios are represented by top-level children of the notations element. Dynamics, ornaments, articulations, and technical indications specific to particular instruments are also top-level children of the notations element. A staccato mark would then be placed within the articulations element.

The tied element represents the visual part of a tie, and the tuplet element represents the visual part of a tuplet. The tie element affects the sound, and the time-modification affects placement, but the tied and tuplet elements indicate that there is something to see on the score indicating the tie or tuplet. (With ties, the two nearly always go together, but with tuplets this is not the case.) The second E-flat in measure 3 of the voice part, which is the end of a tie and start of a tuplet, is represented as:

```
<note>
  <pitch>
    <step>E</step>
    <alter>-1</alter>
    <octave>5</octave>
  </pitch>
  <duration>4</duration>
  <tie type="stop"/>
  <voice>1</voice>
  <type>eighth</type>
  <time-modification>
    <actual-notes>3</actual-notes>
    <normal-notes>2</normal-notes>
  </time-modification>
  <stem>down</stem>
  <notations>
    <tied type="stop"/>
    <tuplet type="start" placement="above" bracket="no"/>
  </notations>
```

```
</note>
```

Slur, tied, and tuplet elements all have a number attribute to distinguish overlapping graphical elements.

Full details for all the different notations can be found in their definitions in the note.dtd file.

## Multi-Part Music

MusicXML contains two elements to help distinguish what is happening in multi-part music: the voice and staff elements.

A staff element should be used wherever possible in multi-staff music like piano parts. Note, forward, and direction elements can all include a staff element. The first cross-staff chord in measure 3 of the piano part is represented as:

```
<note>
  <pitch>
    <step>A</step>
    <octave>3</octave>
  </pitch>
  <duration>1</duration>
  <voice>1</voice>
  <type>eighth</type>
  <accidental>natural</accidental>
  <stem>up</stem>
  <staff>2</staff>
  <beam number="1">begin</beam>
</note>
<note>
  <chord/>
  <pitch>
    <step>C</step>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
  <voice>1</voice>
  <type>eighth</type>
  <stem>up</stem>
  <staff>2</staff>
</note>
<note>
  <chord/>
  <pitch>
    <step>E</step>
    <alter>-1</alter>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
  <voice>1</voice>
  <type>eighth</type>
  <stem>up</stem>
  <staff>1</staff>
</note>
<note>
  <chord/>
  <pitch>
    <step>G</step>
    <octave>4</octave>
```

```
      </pitch>
      <duration>1</duration>
      <voice>1</voice>
      <type>eighth</type>
      <stem>up</stem>
      <staff>1</staff>
    </note>
```

The voice element helps keep track of multiple independent voice parts. Specifying the voice makes it much easier to import MusicXML files into other programs that handle multiple voices, such as Finale with its layer feature. In the piano chord above, having all the notes be part of a chord in voice 1 but with different staff elements ensures that this will be represented as a cross-staff chord. After completing the six chords in the right-hand part, the left-hand chord is represented as a chord in voice 2, using:

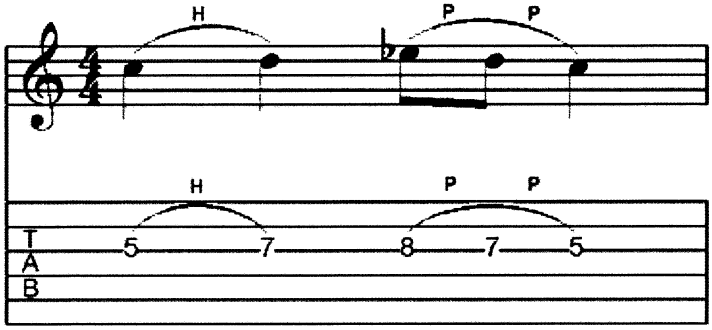
```
    <backup>
      <duration>6</duration>
    </backup>
    <note>
      <pitch>
        <step>F</step>
        <octave>1</octave>
      </pitch>
      <duration>6</duration>
      <voice>2</voice>
      <type>half</type>
      <dot/>
      <stem>down</stem>
      <staff>2</staff>
    </note>
    <note>
      <chord/>
      <pitch>
        <step>F</step>
        <octave>2</octave>
      </pitch>
      <duration>6</duration>
      <voice>2</voice>
      <type>half</type>
      <dot/>
      <stem>down</stem>
      <staff>2</staff>
    </note>
```

# Tablature in MusicXML

Tablature notation provides a direct guideline to the strings and frets used to play music on a guitar or other fretted instrument, often at the expense of precise rhythmic information. In MusicXML, this rhythmic information needs to be specified, although the display is likely to be hidden on a tab part. The main things that need to be added are the fret and string information, the details of the how the strings are tuned, and techniques specific to guitars and other related instruments.

## Fret and String

Here is a simple one-measure guitar part example that we will use to illustrate the basic MusicXML tablature features, using the standard 6-string guitar tuning:




The fret and string information needed to generate tablature for guitar and other stringed instruments is handled the same way as technical indications for other instruments, such as piano fingerings and violin bowings. The technical element contains these types of notations, and two of its component elements represent the note's fret and string. Frets are numbered starting at 0 for the open string. Strings are numbered starting at 1 for the highest string on the instrument.

The fret and string for first note in this example are represented using:

```
<technical>
  <string>3</string>
  <fret>5</fret>
</technical>
```

## String Tuning

An attributes element may include a staff-details element to specify the details of a tab staff. The staff-lines element specifies the number of lines on a tablature staff, usually one for each string. Staff tunings are described with the staff-tuning and capo elements. TAB is one of the values available for clef elements.

The tab part in our example begins with the following attributes:

```
<attributes>
  <divisions>2</divisions>
  <clef>
    <sign>TAB</sign>
    <line>5</line>
  </clef>
```

```
  <staff-details>
    <staff-lines>6</staff-lines>
    <staff-tuning line="1">
      <tuning-step>E</tuning-step>
      <tuning-octave>2</tuning-octave>
    </staff-tuning>
    <staff-tuning line="2">
      <tuning-step>A</tuning-step>
      <tuning-octave>2</tuning-octave>
    </staff-tuning>
    <staff-tuning line="3">
      <tuning-step>D</tuning-step>
      <tuning-octave>3</tuning-octave>
    </staff-tuning>
    <staff-tuning line="4">
      <tuning-step>G</tuning-step>
      <tuning-octave>3</tuning-octave>
    </staff-tuning>
    <staff-tuning line="5">
      <tuning-step>B</tuning-step>
      <tuning-octave>3</tuning-octave>
    </staff-tuning>
    <staff-tuning line="6">
      <tuning-step>E</tuning-step>
      <tuning-octave>4</tuning-octave>
    </staff-tuning>
  </staff-details>
</attributes>
```

## Hammer-ons and Pull-offs

Contemporary guitar notation contains many elements idiomatic to the guitar (and specifically the electric guitar). While elements like harmonics and bends remain in the untested part of MusicXML, the hammer-on and pull-off are both supported in current software.

These elements are represented as technical indications that often go together with a notated slur. In the first half of the bar, a single hammer-on goes together with a single slur:

```
<note>
  <pitch>
    <step>C</step>
    <octave>4</octave>
  </pitch>
  <duration>2</duration>
  <voice>1</voice>
  <type>quarter</type>
  <notations>
    <technical>
      <string>3</string>
      <fret>5</fret>
      <hammer-on type="start" number="1">H</hammer-on>
    </technical>
    <slur type="start" number="1"/>
  </notations>
</note>
<note>
  <pitch>
    <step>D</step>
    <octave>4</octave>
```

```
    </pitch>
    <duration>2</duration>
    <voice>1</voice>
    <type>quarter</type>
    <notations>
      <technical>
        <string>3</string>
        <fret>7</fret>
        <hammer-on type="stop" number="1"/>
      </technical>
      <slur type="stop" number="1"/>
    </notations>
  </note>
```

But in the second half of the bar, a single slur goes together with two pull-offs:

```
  <note>
    <pitch>
      <step>E</step>
      <alter>-1</alter>
      <octave>4</octave>
    </pitch>
    <duration>1</duration>
    <voice>1</voice>
    <type>eighth</type>
    <accidental>flat</accidental>
    <notations>
      <technical>
        <string>3</string>
        <fret>8</fret>
        <pull-off type="start" number="1">P</pull-off>
      </technical>
      <slur type="start" number="1"/>
    </notations>
  </note>
  <note>
    <pitch>
      <step>D</step>
      <octave>4</octave>
    </pitch>
    <duration>1</duration>
    <voice>1</voice>
    <type>eighth</type>
    <notations>
      <technical>
        <string>3</string>
        <fret>7</fret>
        <pull-off type="stop" number="1"/>
        <pull-off type="start" number="2">P</pull-off>
      </technical>
    </notations>
  </note>
  <note>
    <pitch>
      <step>C</step>
      <octave>4</octave>
    </pitch>
    <duration>2</duration>
    <voice>1</voice>
    <type>quarter</type>
    <notations>
```

```
      <technical>
        <pull-off type="stop" number="2"/>
        <string>3</string>
        <fret>5</fret>
      </technical>
      <slur type="stop" number="1"/>
    </notations>
  </note>
```

# Percussion in MusicXML

Percussion instruments can be either pitched or unpitched. Percussion instruments with definite pitch, such as timpani or mallet instruments, are handled as using normal musical notation.

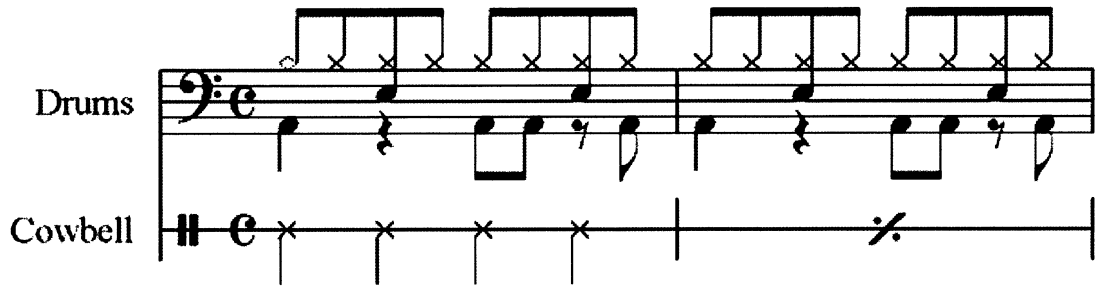
When the instrument has no definite pitch, like a bass drum or snare drum, notation gets stretched. The use of vertical space to represent pitch instead is used to represent different instruments. Yet sometimes we have more percussion instruments handled by one player than can fit into the lines on a staff. In other cases, we may have only one unpitched instrument to notate, where the vertical space on a 5-line staff would be wasted.

MusicXML allows percussion music to be represented both as heard and as notated by use of several features:

- Representing unpitched instruments visually by specifying placement on the staff.
- As with tablature, staves may be more or less than 5 lines high.
- Specifying multiple instruments per part
- Specifying the MIDI playback for each instrument (for instance, by a single MIDI pitch in a percussion kit).
- Alternate notehead shapes let multiple instruments to share a single line on the staff.

## Unpitched Notes

To illustrate MusicXML's percussion features, here's a two-bar example for two players: one on a drum kit, and one on cowbell:




In the drum part, the top space (B in bass clef) is used for the cymbal (diamond notehead) and hi-hat (x notehead). The E space is used for the snare drum, and the bottom A space is used for the bass drum. The cowbell player has only one instrument, so it is represented on a one-line staff.

Since these notes have no definite pitch, it would be misleading to represent them using the pitch element. An analysis program looking for a series of repeated B's should not return this piece of music. Nor should a program looking for a series of repeated F-sharps, based on the General MIDI pitch for a closed hi-hat.

Instead, we represent percussion and other unpitched instruments with the unpitched element. As with rests, there are display-step and display-octave elements to indicate where the note should

appear on the staff, based on the current clef. So the cymbal and hi-hat notes on this bass clef staff are represented as:

```
<unpitched>
  <display-step>B</display-step>
  <display-octave>3</display-octave>
</unpitched>
```

Percussion clef is treated like treble clef when determining the display-step and display-octave. So if this part had been notated in percussion clef instead of bass clef, the resulting MusicXML would be:

```
<unpitched>
  <display-step>G</display-step>
  <display-octave>5</display-octave>
</unpitched>
```

## Staff Lines

The cowbell part is an example where one player has one instrument, so the part is notated on one line both for clarity and saving space. The single line is specified within the attributes element for the part:

```
<attributes>
  <divisions>1</divisions>
  <key>
    <fifths>0</fifths>
    <mode>major</mode>
  </key>
  <time symbol="common">
    <beats>4</beats>
    <beat-type>4</beat-type>
  </time>
  <clef>
    <sign>Percussion</sign>
  </clef>
  <staff-details>
    <staff-lines>1</staff-lines>
  </staff-details>
</attributes>
```

How do we determine the display-step and display-octave for a one-line staff? Since the percussion clef is treated like a treble clef, the G in octave 4 is on the second line of the staff. For this one-line staff, that G is one imaginary line above the staff (for a 2-line staff, it's the top line, and the middle line on a 3-line staff). Therefore the unpitched elements for the cowbell part are:

```
<unpitched>
  <display-step>E</display-step>
  <display-octave>4</display-octave>
</unpitched>
```

## Multiple Instruments Per Part

Percussion parts are one case of many where multiple instruments are sharing the same part. This is represented in MusicXML by including more than one score-instrument element within a score-part. Each note can then contain an instrument element that refers back to the id of the score-instrument. Note that the score-instrument id must be unique throughout the entire piece. Two score-part elements cannot each have a score-instrument element with the same id.

To represent MIDI playback for each instrument, we add a midi-instrument element for each score-instrument. In this case, we are using a General MIDI instrument, so we include a midi-channel of 10 for each instrument. We then use the midi-unpitched element to indicate the MIDI pitch that corresponds to a particular sound in a General MIDI drum kit.

The part-list for our two-bar example looks like this:

```
<part-list>
  <score-part id="P1">
    <part-name>Drums</part-name>
    <score-instrument id="P1-X2">
      <instrument-name>Bass Drum</instrument-name>
    </score-instrument>
    <score-instrument id="P1-X4">
      <instrument-name>Snare</instrument-name>
    </score-instrument>
    <score-instrument id="P1-X6">
      <instrument-name>Hi-Hat Closed</instrument-name>
    </score-instrument>
    <score-instrument id="P1-X13">
      <instrument-name>Crash</instrument-name>
    </score-instrument>
    <midi-instrument id="P1-X2">
      <midi-channel>10</midi-channel>
      <midi-program>1</midi-program>
      <midi-unpitched>37</midi-unpitched>
    </midi-instrument>
    <midi-instrument id="P1-X4">
      <midi-channel>10</midi-channel>
      <midi-program>1</midi-program>
      <midi-unpitched>39</midi-unpitched>
    </midi-instrument>
    <midi-instrument id="P1-X6">
      <midi-channel>10</midi-channel>
      <midi-program>1</midi-program>
      <midi-unpitched>43</midi-unpitched>
    </midi-instrument>
    <midi-instrument id="P1-X13">
      <midi-channel>10</midi-channel>
      <midi-program>1</midi-program>
      <midi-unpitched>50</midi-unpitched>
    </midi-instrument>
  </score-part>
  <score-part id="P2">
    <part-name>Cowbell</part-name>
    <score-instrument id="P2-X1">
      <instrument-name>Cowbell</instrument-name>
    </score-instrument>
    <midi-instrument id="P2-X1">
      <midi-channel>10</midi-channel>
      <midi-program>1</midi-program>
      <midi-unpitched>57</midi-unpitched>
    </midi-instrument>
  </score-part>
</part-list>
```

Each note then includes an unpitched element to show where the note is on the staff, and an instrument element to indicate which instrument is used and how to play it back using MIDI. The first bass drum note looks like this:

```
<note>
  <unpitched>
    <display-step>F</display-step>
    <display-octave>4</display-octave>
  </unpitched>
  <duration>2</duration>
  <instrument id="P1-X2"/>
  <voice>2</voice>
  <type>quarter</type>
  <stem>down</stem>
</note>
```

Our Dolet for Finale plug-in creates these instrument names like "P1-X2" to make it easy to generate unique names for each score instrument. It would be better to use more readable names that are still unique across the entire piece.

## Notehead Shapes

The one remaining task is to specify the alternate noteheads that distinguish the hi-hat from the cymbal. While the MusicXML playback can distinguish these by the use of different instruments, a drummer will certainly appreciate having different notehead shapes for different instrument on the same line.

The MusicXML notehead element specifies these different shapes. Standard values include slash, triangle, diamond, square, cross, x, circle-x, normal, and none. Enclosed shapes like normal, diamond, triangle, and square can use the filled attribute to indicate a filled or hollow shape. The first two notes of the cymbal/hi-hat line look like this:

```
<note>
  <unpitched>
    <display-step>B</display-step>
    <display-octave>3</display-octave>
  </unpitched>
  <duration>1</duration>
  <instrument id="P1-X13"/>
  <voice>1</voice>
  <type>eighth</type>
  <stem>up</stem>
  <notehead filled="no">diamond</notehead>
  <beam number="1">begin</beam>
</note>
<note>
  <unpitched>
    <display-step>B</display-step>
    <display-octave>3</display-octave>
  </unpitched>
  <duration>1</duration>
  <instrument id="P1-X6"/>
  <voice>1</voice>
  <type>eighth</type>
  <stem>up</stem>
  <notehead>x</notehead>
  <beam number="1">continue</beam>
</note>
```

The filled attribute on the notehead element is also useful for multi-part piano music. There are many cases where, for instance, a downstem half note shares a hollow notehead with an upstem

eighth note. The eighth note can specify that it uses an unfilled normal notehead, making things display correctly when moving back and forth between notation programs.

## Measure Styles

Our cowbell part also shows the use of a one-bar repeat symbol. This and similar repeats and multimeasure rests are represented using the measure-style element. The music in the second bar should be specified just as in the first bar (four notes with display-step E, display-octave 4, and instrument P2-X1). At the start of the bar, we indicate the beginning of the one-measure repeat style:

```
<measure number="2">
  <attributes>
    <measure-style>
      <measure-repeat type="start">1</measure-repeat>
    </measure-style>
  </attributes>
```

Further details about the measure-style element can be found in the attributes.dtd file.

# Advanced Features

This part of the tutorial will describe further advanced MusicXML features, including more precise control of notation formatting, sound, or less-frequently used notation.

Features to be discussed will include:

- New MusicXML 1.1 formatting features
- Placement and orientation
- Position
- Sound attributes
- Nonstandard key signatures

Stop reading here for ECE402 homework.