MusicXML

craig@ccrma.stanford.edu

25 February 2014

Parameters

Fixed — function(int one, int two, int three)
(like MIDI)

C

Optional — function(int one, int two = 2, int three=3)
(like Guido, SCORE)

C++

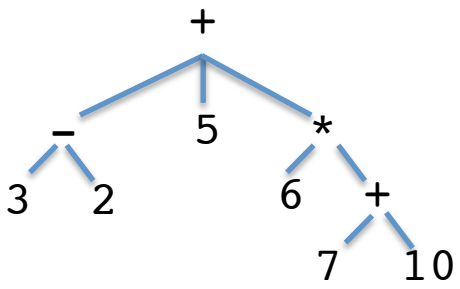
Variable — function(const char* format, …)
http://cc.byexamples.com/2007/01/18/va-list-create-function-like-printf-2

Key — (function :key1 value1 :key2 value2)

Lisp

Tree — (+ (- 3 2) 5 (* 6 (+ 7 10)))

(recursive key system)



MIDI Parameters

- All MIDI protocol parameters are fixed excepts for “system exclusive messages”
- Meta messages (component of MIDI files, not MIDI protocol) are variable.

0x90 60 127 note(channel, key, velocity)

0xE6 0x7f 0x7f bend(channel, LSB, MSB)

- Allows hot-plugging of MIDI cable.
- Limits expandability (function space maximized with fixed parameter commands)

SCORE Parameters

- SCORE items are all variable length fixed parameter lists.
- Similar to MIDI meta message system, but better extensibility
- Identical to Music V (C Sound) parameter system

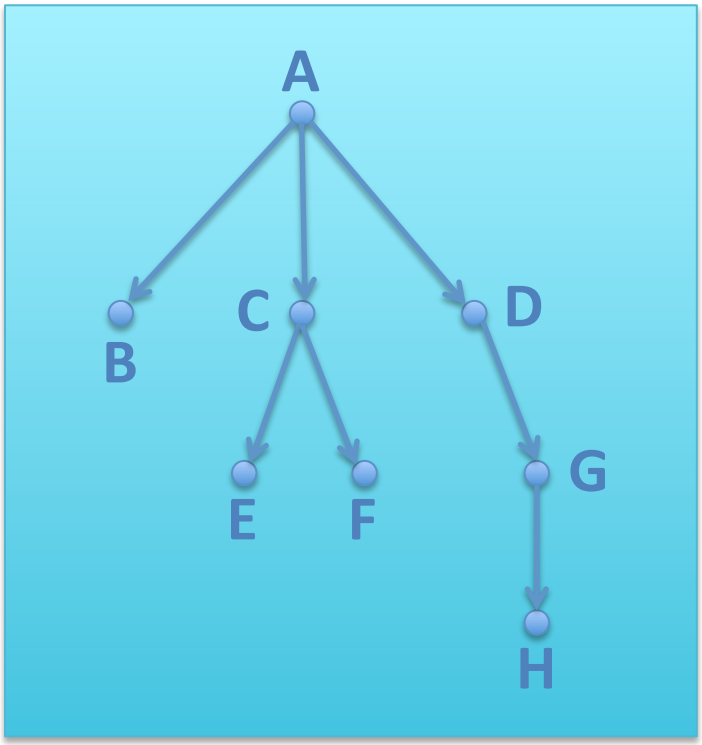
http://www.csounds.com/chapter1/index.html

```
8  1 0 0 0.6 128.146
14 1 0 3
3  1 1.2 0 0.8
17 1 5.997 0 -1
1  1 9.297 7 20 1 2
1  1 20.566 4 10 2 4
1  1 50.64 8 20 1 2
5  1 50.64 8.5 8.5 64.016 1.579 -2
14 1 61.923 1
1  1 64.016 8 20 1 2
1  1 75.291 6 10 1 2
1  1 86.561 5 10 2 4
14 1 109.113 1
1  1 111.206 9 20 2 4
14 1 128.146 1 3
```

- Allows for both forwards and backwards compatibility:
  - New parameters added to end of current list
  - Old program ignores (but preserves) unknown parameters.

XML

- XML describes a tree structure:



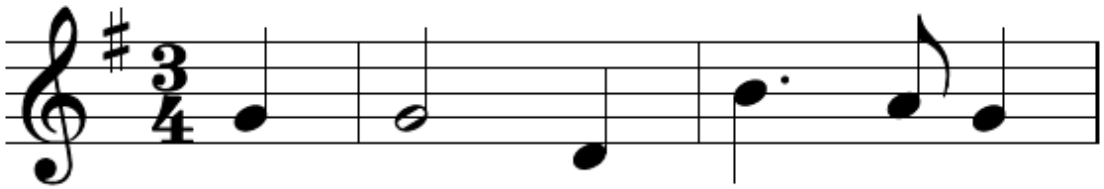
- Serialization showing hierarchy:

```
<A>
   <B/>
   <C>
      <E/>
      <F/>
   </C>
   <D>
      <G>
         <H/>
      </G>
   </D>
</A>
```

- Equivalent serialization: <A><B/><C><E/><F/></C><D><G><H/></G></D></A>
- LISP analogy: (a b (c e f) (d (g h)))

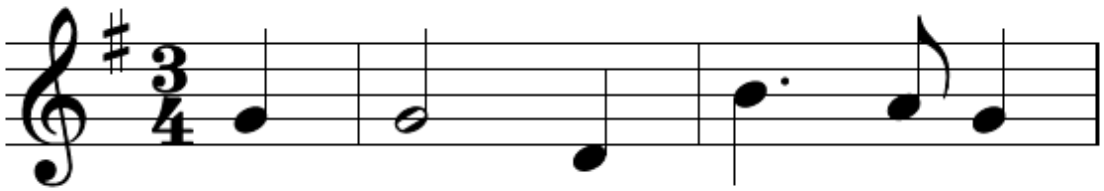
Non-XML data trees

- SharpEye uses a form of tree structure for its data
- LISP-based ENP music editor uses tree structure:



```
(:begin :score
      (:begin :part1
            :staff :treble-staff
            :key-signature :g-major
            :time-signature (3 4)
            (:begin :voice1
                  (
                        :time-signature (3 4 :kind :pickup)
                        (1 ((1 :notes (67))))
                  )
                  (:begin :measure1
                        (2 ((1 :notes (67))))
                        (1 ((1 :notes (74))))
                  )
                  (:begin :measure2
                        (2 (
                              (3 :notes (71))
                              (1 :notes (69))
                        ))
                        (1 ((1 :notes (67))))
                  )
                  (:begin :measure3
                        (2 (
                              (3 :notes (67))
                              (1 :notes (69))
                        ))
                        (1 ((1 :notes (71))))
                  )
```

XML as a container for non-tree data



```
8  1   0.000 0  0 100
3  1   1.500
17 1   9.444 0  1
18 1  13.444 0  3   4
1  1  20.944 5 10   0 1.0
14 1  32.290 1
1  1  35.679 5 10   1 2.0
1  1  52.032 2 10   0 1.0
14 1  63.378 1
1  1  66.767 7 20   0 1.5 0 10
1  1  80.853 6 10   0 0.5 0  1
1  1  88.654 5 10   0 1.0
14 1 100.000 1
```

```
<SCORE version="4">
  <item p1="8" p2="1" p6="100" />
  <item p1="3" p2="1" p3="1.5" />
  <item p1="17" p2="1" p3="9.444" p5="1" />
  <item p1="18" p2="1" p3="13.444" p5="3" p6="4" />
  <item p1="1" p2="1" p3="20.944" p4="5" p5="10" p7="1" />
  <item p1="14" p2="1" p3="32.29" p4="1" />
  <item p1="1" p2="1" p3="35.679" p4="5" p5="10" p6="1" p7="2" />
  <item p1="1" p2="1" p3="52.032" p4="2" p5="10" p7="1" />
  <item p1="14" p2="1" p3="63.378" p4="1" />
  <item p1="1" p2="1" p3="66.767" p4="7" p5="20" p7="1.5" p9="10" />
  <item p1="1" p2="1" p3="80.853" p4="6" p5="10" p7="0.5" p9="1" />
  <item p1="1" p2="1" p3="88.654" p4="5" p5="10" p7="1" />
  <item p1="14" p2="1" p3="100" p4="1" />
</SCORE>
```

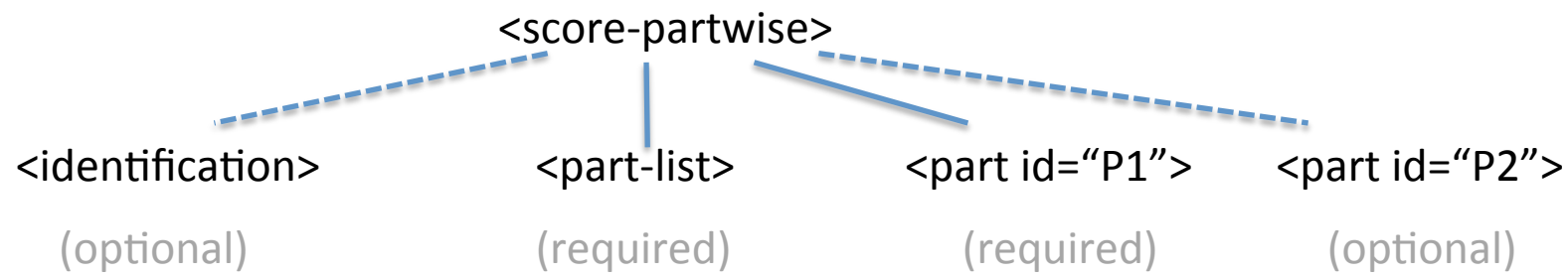
XML

- Advantage: Simple parsing model for data storage
  - Like MIDI, SCORE, LISP, Humdrum
  - Unlike Guido, Lilypond, C, C++, Java, JavaScript (lex/bison type formats)

- Allows for hierarchical structuring of data
  - **Good**: music notation usually fits well into hierarchical model
    - Useful for manipulating music
  - **Bad**: music notation is 2-dimensional, XML is 1-dimensional (superposition of multiple hierarchies)

- Allows for forwards compatibility, and backwards compatibility if careful
  - Possible to add new parameters without altering parsing

MusicXML Data hierarchy (root)




```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 1.0 Partwise//EN"
                                "http://www.musicxml.org/dtds/1.0/partwise.dtd">
<score-partwise>
```

<score-partwise> is the *root element*

```
<!ELEMENT score-partwise (%score-header;, part+)>

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

DTD/Schema

DTD

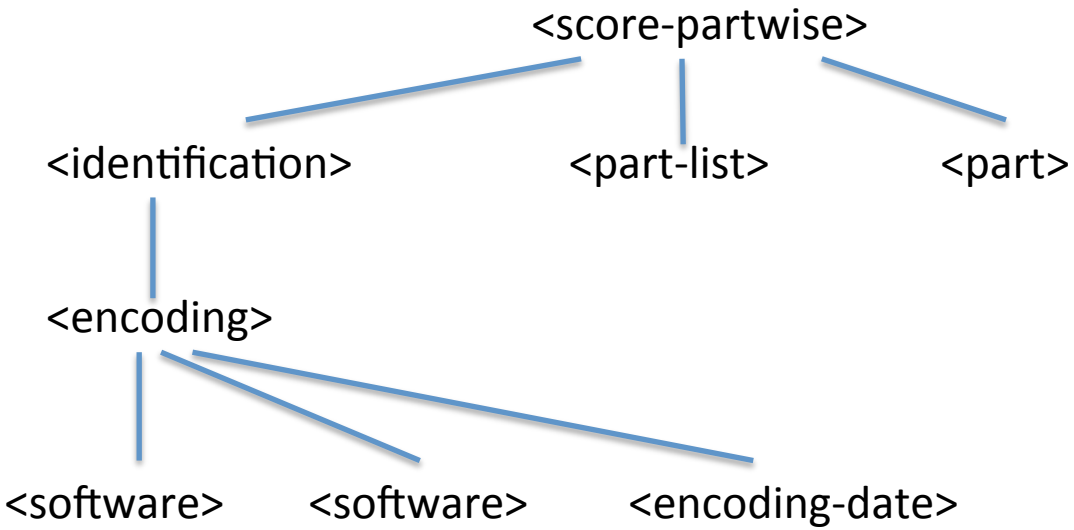
```
<!ELEMENT score-partwise (%score-header;, part+)>

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

Schema

```
<xs:element name="score-partwise" block="extension substitution" final="#all">
 <xs:annotation>
  <xs:documentation>
  The score-partwise element is the root element for a partwise
  MusicXML score. It includes a score-header group followed by a
  series of parts with measures inside. The document-attributes
  attribute group includes the version attribute.
  </xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:sequence>
   <xs:group ref="score-header"/>
   <xs:element name="part" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="measure" maxOccurs="unbounded">
       <xs:complexType>
        <xs:group ref="music-data"/>
        <xs:attributeGroup ref="measure-attributes"/>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attributeGroup ref="part-attributes"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="document-attributes"/>
 </xs:complexType>
</xs:element>
```

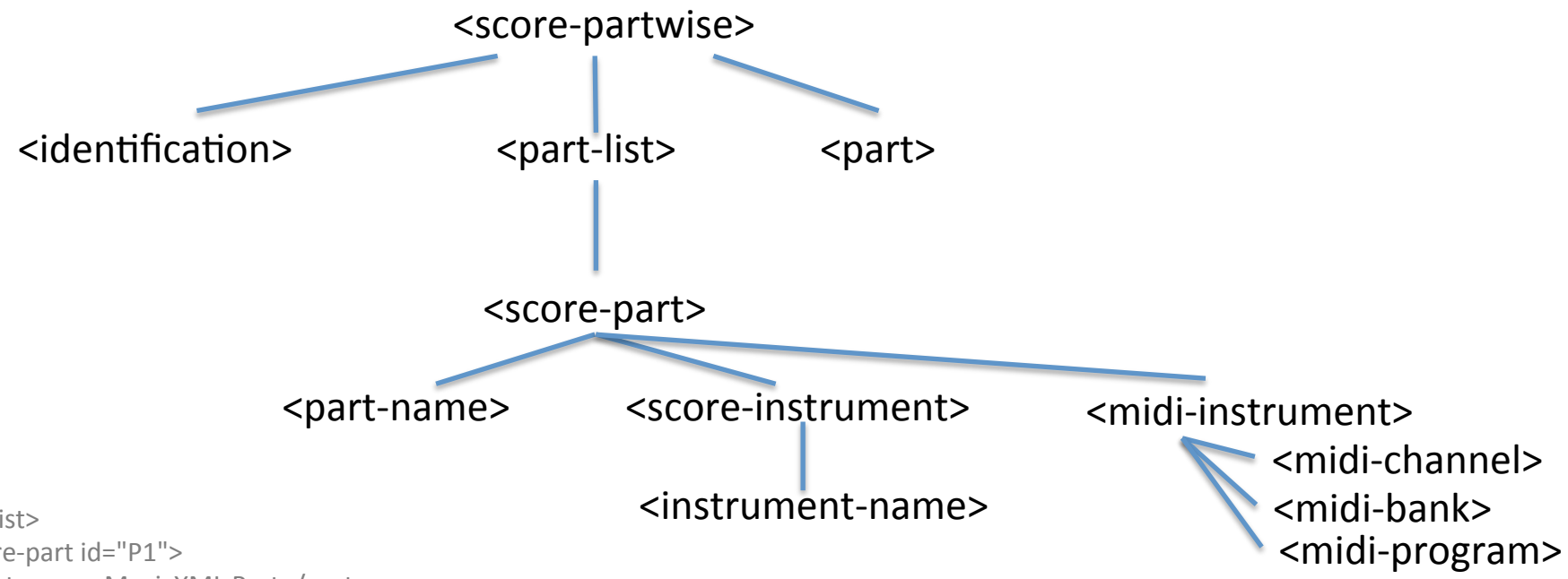
http://www.musicxml.com/UserManuals/MusicXML/MusicXML.htm

MusicXML Data hierarchy (header)



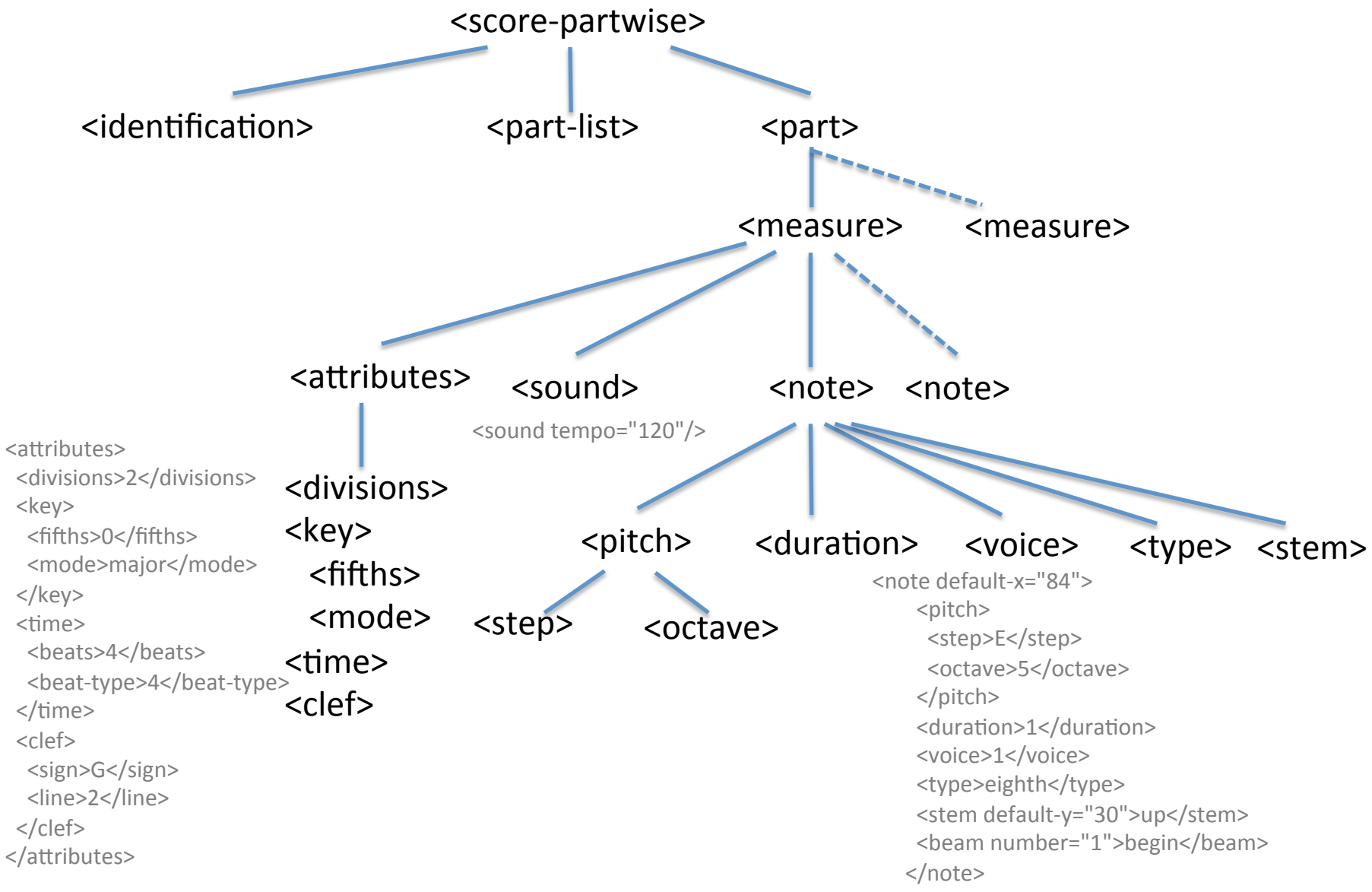

```
<identification>
   <encoding>
     <software>Finale 2012 for Mac</software>
     <software>Dolet Light for Finale 2012</software>
     <encoding-date>2013-02-25</encoding-date>
   </encoding>
 </identification>
```

MusicXML Data hierarchy (header 2)




```
<part-list>
  <score-part id="P1">
   <part-name>MusicXML Part</part-name>
   <score-instrument id="P1-I1">
    <instrument-name>Garritan: ARIA Player</instrument-name>
   </score-instrument>
   <midi-instrument id="P1-I1">
    <midi-channel>1</midi-channel>
    <midi-bank>15489</midi-bank>
    <midi-program>1</midi-program>
   </midi-instrument>
  </score-part>
 </part-list>
```

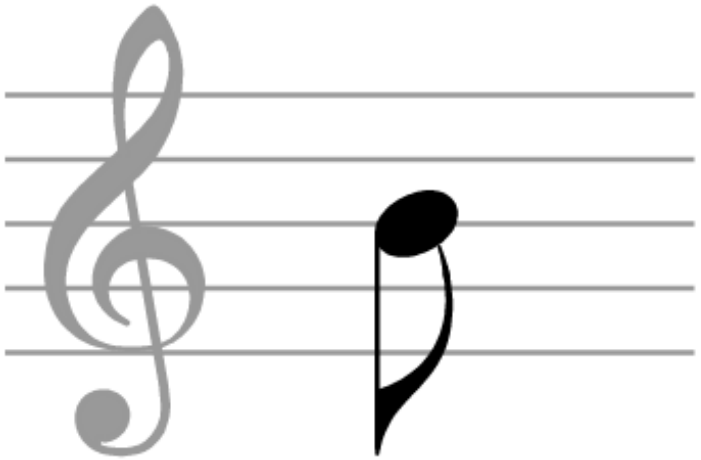
MusicXML Data hierarchy (part)



MusicXML <note>

http://www.musicxml.com/UserManuals/MusicXML/MusicXML.htm#EL-MusicXML-note.htm

```
<note>
      <pitch>
        <step>B</step>
        <octave>4</octave>
      </pitch>
      <duration>16</duration>
      <voice>1</voice>
      <type>eighth</type>
      <stem default-y="-50">down</stem>
  </note>
```



MuseData “<note>”:

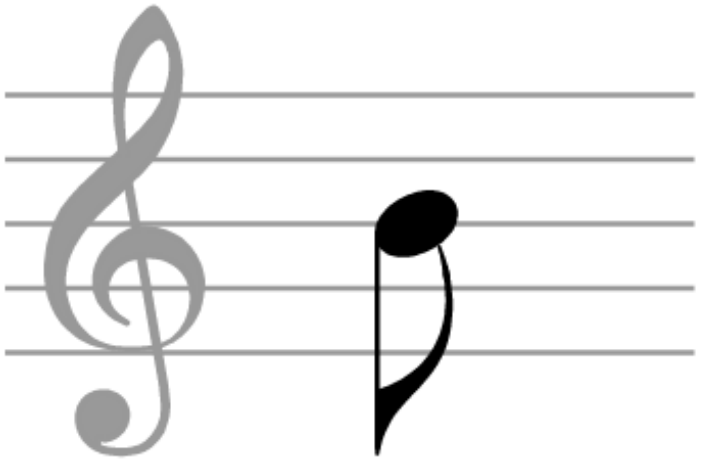
```
                1         2         3
12345678901234567890123456789
B4      1       1 e     d
```

<pitch> <step> <octave> <duration> <voice> <type> <stem>

MusicXML <note>

http://www.musicxml.com/UserManuals/MusicXML/MusicXML.htm#EL-MusicXML-note.htm

```
<note>
      <pitch>
        <step>B</step>
        <octave>4</octave>
      </pitch>
      <duration>16</duration>
      <voice>1</voice>
      <type>eighth</type>
      <stem default-y="-50">down</stem>
  </note>
```

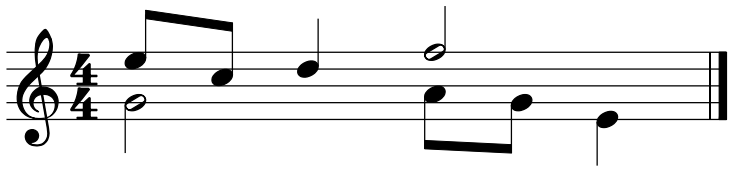


MuseData “<note>”:

```
                1         2         3
12345678901234567890123456789012345678901234567890
B4      1       1 e       d
P C23:y-50
```

Column 23

MusicXML Voices/Layers



MusicXML

```
<measure>
  <attributes>
  <note> E5, 1 tick
  <note> C5, 1 tick
  <note> D5, 2 ticks
  <note> F5, 4 ticks
  <backup> 8 ticks
  <note> G4, 4 ticks
  <note> A4, 1 tick
  <note> G4, 1 tick
  <note> E4, 2 tick
</measure>
```

Voice 1:

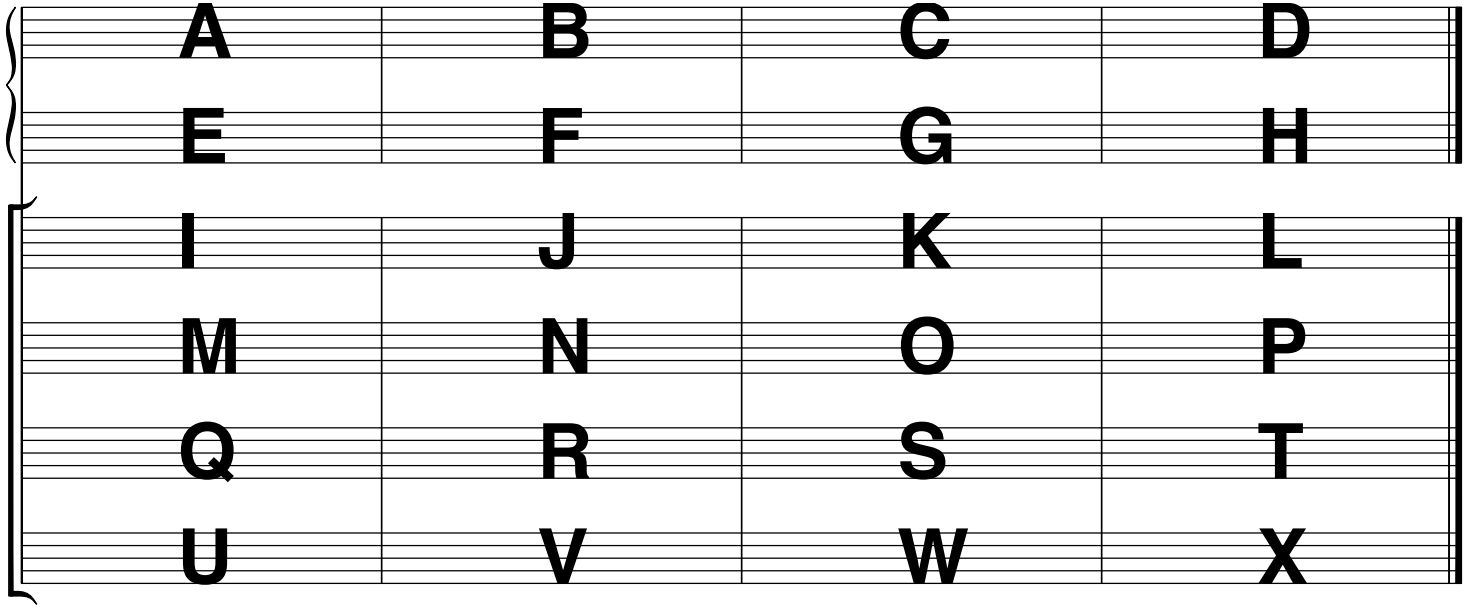
Voice 2:

MuseData

```
Group memberships: score
score: part 1 of 1
$    Q:2 K:0 T:1/1 C:4
E5       1        1 e      u   [
C5       1        1 e      u   ]
D5       2        1 q      u
F5       4        1 h      u
back     8
G4       4        2 h      d
A4       1        2 e      d   [
G4       1        2 e      d   ]
E4       2        2 q      d
mheavy2
/END
```

Partwise/timewise

- <score-partwise> stores score one part (staff) sequentially (part->measure)
- <score-timewise> score each measure sequentially for all parts (measure->part)
- <score-timewise> is about as common as MIDI Type-2 files.
- <score-timewise> is a quasi-realtime encoding (not strictly real-time).



- <score-partwise>: ABCD, EFGH, IJKL, MNOP, QRST, UVWX
- <score-timewise>: AEIMQU, BFJNRV, CGKOSW, DHLPTX

- <opus>: multiple movements of (partwise or timewise).

MusicXML versions

http://www.musicxml.com
http://en.wikipedia.org/wiki/MusicXML

MusicXML 1.0	2004
MusicXML 1.1	2005
MusicXML 2.0	2007
MusicXML 3.0	2011

MusicXML 3.0

- Compressed MusicXML: (.mxl): ZIP file which can include linked material as well as main XML file.

- Standardized list of instruments
  - http://www.musicxml.com/dtds/3.0/sounds.xml
  - http://www.humdrum.org/Humdrum/guide.append2.html

- Jianpu notation, microtonal music (Turkish music), AlphaNotes
  - http://bennyt85erhu.wordpress.com/jianpu
  - http://www.hinesmusic.com/What_Are_Makams.html
  - http://blog.finalemusic.com/post/2011/10/20/Finale-Quick-Tips-AlphaNotes.aspx

- More graphic symbol representations for perussion, handbells, haupt-, nebenstimme
  - http://en.wikipedia.org/wiki/Hauptstimme

Data Interchange Cases