Rhythm and Clues – User's Guide

Introduction

This is a program to take a file in MusicXML[1] format[2] and modify the duration of its notes according to a user-specified pattern. The output file can then be re-imported to a music-notation program and then played by the artist.

The goal is to help the musician understand how to play a difficult passage by playing it in different rhythms[3].

The program is written in C# so it can be ported readily to Windows, MacOS and Linux. It's a console app[4] as using the program wouldn't benefit much from a GUI interface and cross-platform GUI libraries are notoriously limited and/or complex to program and/or expensive.

Running the Program Under Windows

Open a DOS box and navigate to the directory where the RhythmAndClues program resides. Simply type

```
RhythmAndClues program.txt
```

where program.txt is the input to the RhythmAndClues program[5], described below.

Running the Program Under MacOS or Linux

Here's where it gets just a bit complicated. A touch of history – When C# and the .Net Framework[6] first appeared in June 2000, it was back in the Steve Balmer days, when *everything* was focused on Windows. But these days, Microsoft is much more open about things, and wants to support many of its products on multiple platforms.

".Net Core" is a large subset of the original .Net Framework. Subroutines that only apply to a Windows environment[7] have been stripped out and the rest have been ported to the MacOS and Linux platforms.

To run RhythmAndClues on a Mac, you must first install .Net Core (the current version (March 2019) is 2.2, but 3.0 is currently in beta). The program has been tested on Windows under .Net Core version 2.1, but any later (non-beta) versions should work perfectly.

To be able to (among other things) compile C# programs and run them, .Net Core also includes the "dotnet" command. Just as to compile and/or run a Java program you have to use the "java" program (e.g. write "java myprog") you have to use the "dotnet" command.

I don't have a Mac, so have never tested the following, but my current best guess is that it would work somewhat as follows:

- Create a folder with the same name as the program (i.e. "RhythmAndClues")
- Inside it place the *.cs files that comprise the RhythmAndClues source modules
- Run "dotnet build" to compile the program

[1] https://www.musicxml.com/
[2] It's been tested only on version 3, and mostly on files from *Finale* (https://www.finalemusic.com/)
[3] Did I get that right?
[4] Also known as a terminal app.
[5] Prefix the filename with a path, if you want
[6] A very large class/subroutine library including (among many others) routines for string manipulation, Internet access, graphics routines, and so on.
[7] Unfortunately this includes graphics routines.

- Create/modify a file with the user input file (see next section). It can be named anything, but let's assume it's called "program.txt".
- When it's compiled cleanly, run "dotnet run program.txt"

For more information on how to install .Net Core on Windows/Mac, see

- https://dotnet.microsoft.com/download -- You need download only the SDK (Software Development Kit). It already contains the .Net Core Runtime component.
  - Also note the Windows / Linux / macOS links near the top
  - And check out the *Hello world in 10 minutes tutorial* link above that.

Program Input

The program takes as input a simple text file[8] with commands described below. But first a few general comments:

- % -- Marks the start of a comment. It and the rest of the line are ignored
- All empty lines (with just blanks and/or tabs) are ignored
- Commands are case-insensitive – "Load", "LOAD", "LoaD", etc. are equivalent
- Filenames -- Do not put quotes around filenames, even if they contain blanks. However multiple blanks will be reduced to a single blank
- Some commands depend on previous commands being issued. For example, a SAVE command before a LOAD command is considered an error.

The main commands are:

- LOAD <filename> -- Loads an XML file into memory
- APPLY-PATTERN <list of durations> -- Applies the pattern to the note from the previous LOAD
- SAVE <filename> -- Saves the result to the specified filename. The filename must not be the same as the one specified in the LOAD command (this is a safety feature). Also, if the output file exists, you'll be prompted to ask if you want the output file overwritten.
- HELP – shows a summary of the available commands

Secondary commands are mainly for testing during development and include:

- DUMP-MEASURES – Shows a formatted listing of relevant parts of the most recent LOADed file.
- DELETE-MEASURES – Deletes all measures in the input file. Suitable for testing inserting a new set of measures
- RESTORE-MEASURES – Puts back the original measures

Sample Input file

A valid input file could say

[8] Created in, for example, Notepad in Windows. Do not use any kind of word processor that would let you format the document with, say, boldface, or change the font size of the text and so on.

```
% Help

Load ../../../XmlFiles/input A major scale.xml
Dump-Measures
Apply-Pattern  18 6 18 6 6 6
Dump-Measures
Save ../../../XmlFiles/input A major scale-LRS.xml
```

Sample Output:

```
Rhythm And Clues -- version 2019-02-28, 08:09 PM

First pass -- checking syntax.
[1] >>     % Help
[2] >>
[3] >>     Load ../../../XmlFiles/input A major scale.xml
[4] >>     Dump-Measures
[5] >>     Apply-Pattern  18 6 18 6 6 6
[6] >>     Dump-Measures
[7] >>     Save ../../../XmlFiles/input A major scale-LRS.xml
The output filename exists.
Enter Y to overwrite or N to quit:
[8] >>

Syntax looks good. Running the program...

[1] >>     % Help

[2] >>

[3] >>     Load ../../../XmlFiles/input A major scale.xml

[4] >>     Dump-Measures

======================================
Measure[1]      Pitch  Duration    Type
                -----  --------  ------
   [1]             A3         1  eighth
   [2]            G13         1  eighth
   [3]             A3         1  eighth
   [4]             B3         1  eighth
   [5]            C14         1  eighth
   [6]             D4         1  eighth
   [7]             E4         1  eighth
   [8]            F14         1  eighth

======================================
Measure[2]      Pitch  Duration    Type
                -----  --------  ------
   [1]            G14         1  eighth
   [2]             A4         1  eighth
   [3]             B4         1  eighth
   [4]            C15         1  eighth
   [5]             D5         1  eighth
   [6]             E5         1  eighth
   [7]           F15         1  eighth
   [8]           G15         1  eighth

======================================
Measure[3]      Pitch  Duration    Type
                -----  --------  ------
   [1]            A5         1  eighth
   [2]            B5         1  eighth
   [3]           C16         1  eighth
   [4]            D6         1  eighth
   [5]            E6         1  eighth
   [6]           F16         1  eighth
   [7]           G16         1  eighth
   [8]            A6         1  eighth

======================================
Measure[4]      Pitch  Duration    Type
                -----  --------  ------
   [1]           G16         1  eighth
   [2]           F16         1  eighth
   [3]            E6         1  eighth
   [4]            D6         1  eighth
   [5]           C16         1  eighth
   [6]            B5         1  eighth
   [7]            A5         1  eighth
   [8]           G15         1  eighth

======================================
Measure[5]      Pitch  Duration    Type
                -----  --------  ------
   [1]           F15         1  eighth
   [2]            E5         1  eighth
   [3]            D5         1  eighth
   [4]           C15         1  eighth
   [5]            B4         1  eighth
   [6]            A4         1  eighth
   [7]           G14         1  eighth
   [8]           F14         1  eighth

======================================
Measure[6]      Pitch  Duration    Type
                -----  --------  ------
   [1]            E4         1  eighth
   [2]            D4         1  eighth
   [3]           C14         1  eighth
   [4]            B3         1  eighth
   [5]            A3         1  eighth
   [6]           C14         1  eighth
   [7]            B3         1  eighth
   [8]            A3         1  eighth

[5] >>     Apply-Pattern  18 6 18 6 6 6
Converted note[1] duration/type from 1/eighth to 18/eighth
Converted note[2] duration/type from 1/eighth to 6/16th
Converted note[3] duration/type from 1/eighth to 18/eighth
Converted note[4] duration/type from 1/eighth to 6/16th
Converted note[5] duration/type from 1/eighth to 6/16th
Converted note[6] duration/type from 1/eighth to 6/16th
Converted note[7] duration/type from 1/eighth to 18/eighth
Converted note[8] duration/type from 1/eighth to 6/16th
Converted note[9] duration/type from 1/eighth to 18/eighth
Converted note[10] duration/type from 1/eighth to 6/16th
Converted note[11] duration/type from 1/eighth to 6/16th
Converted note[12] duration/type from 1/eighth to 6/16th
Converted note[13] duration/type from 1/eighth to 18/eighth
Converted note[14] duration/type from 1/eighth to 6/16th
Converted note[15] duration/type from 1/eighth to 18/eighth
Converted note[16] duration/type from 1/eighth to 6/16th
Converted note[17] duration/type from 1/eighth to 6/16th
Converted note[18] duration/type from 1/eighth to 6/16th
Converted note[19] duration/type from 1/eighth to 18/eighth
Converted note[20] duration/type from 1/eighth to 6/16th
Converted note[21] duration/type from 1/eighth to 18/eighth
Converted note[22] duration/type from 1/eighth to 6/16th
Converted note[23] duration/type from 1/eighth to 6/16th
Converted note[24] duration/type from 1/eighth to 6/16th
Converted note[25] duration/type from 1/eighth to 18/eighth
Converted note[26] duration/type from 1/eighth to 6/16th
Converted note[27] duration/type from 1/eighth to 18/eighth
Converted note[28] duration/type from 1/eighth to 6/16th
Converted note[29] duration/type from 1/eighth to 6/16th
Converted note[30] duration/type from 1/eighth to 6/16th
Converted note[31] duration/type from 1/eighth to 18/eighth
Converted note[32] duration/type from 1/eighth to 6/16th
Converted note[33] duration/type from 1/eighth to 18/eighth
Converted note[34] duration/type from 1/eighth to 6/16th
Converted note[35] duration/type from 1/eighth to 6/16th
Converted note[36] duration/type from 1/eighth to 6/16th
Converted note[37] duration/type from 1/eighth to 18/eighth
Converted note[38] duration/type from 1/eighth to 6/16th
Converted note[39] duration/type from 1/eighth to 18/eighth
Converted note[40] duration/type from 1/eighth to 6/16th
Converted note[41] duration/type from 1/eighth to 6/16th
Converted note[42] duration/type from 1/eighth to 6/16th
Converted note[43] duration/type from 1/eighth to 18/eighth
Converted note[44] duration/type from 1/eighth to 6/16th
Converted note[45] duration/type from 1/eighth to 18/eighth
Converted note[46] duration/type from 1/eighth to 6/16th
Converted note[47] duration/type from 1/eighth to 6/16th
Converted note[48] duration/type from 1/eighth to 6/16th
Time signature set to 5 / 8[9]
***Inserting new measure [1] -- 6 notes
      [1]: 18 A3
      [2]:  6 G13
      [3]: 18 A3
      [4]:  6 B3
      [5]:  6 C14
      [6]:  6 D4
```

[9] The algorithm to calculate the time signature isn't currently clear. This line was produced by an earlier version of the program.

```
***Inserting new measure [2] -- 6 notes
      [1]: 18 E4
      [2]:  6 F14
      [3]: 18 G14
      [4]:  6 A4
      [5]:  6 B4
      [6]:  6 C15
***Inserting new measure [3] -- 6 notes
      [1]: 18 D5
      [2]:  6 E5
      [3]: 18 F15
      [4]:  6 G15
      [5]:  6 A5
      [6]:  6 B5
***Inserting new measure [4] -- 6 notes
      [1]: 18 C16
      [2]:  6 D6
      [3]: 18 E6
      [4]:  6 F16
      [5]:  6 G16
      [6]:  6 A6
***Inserting new measure [5] -- 6 notes
      [1]: 18 G16
      [2]:  6 F16
      [3]: 18 E6
      [4]:  6 D6
      [5]:  6 C16
      [6]:  6 B5
***Inserting new measure [6] -- 6 notes
      [1]: 18 A5
      [2]:  6 G15
      [3]: 18 F15
      [4]:  6 E5
      [5]:  6 D5
      [6]:  6 C15
***Inserting new measure [7] -- 6 notes
      [1]: 18 B4
      [2]:  6 A4
      [3]: 18 G14
      [4]:  6 F14
      [5]:  6 E4
      [6]:  6 D4
***Inserting new measure [8] -- 6 notes
      [1]: 18 C14
      [2]:  6 B3
      [3]: 18 A3
      [4]:  6 C14
      [5]:  6 B3
      [6]:  6 A3

[6] >>     Dump-Measures

======================================
Measure[1]      Pitch  Duration    Type
                -----  --------  ------
```

```
   [1]             A3        18  eighth
   [2]            G13         6    16th
   [3]             A3        18  eighth
   [4]             B3         6    16th
   [5]            C14         6    16th
   [6]             D4         6    16th

======================================
Measure[2]      Pitch  Duration    Type
                -----  --------  ------
   [1]             E4        18  eighth
   [2]            F14         6    16th
   [3]            G14        18  eighth
   [4]             A4         6    16th
   [5]             B4         6    16th
   [6]            C15         6    16th

======================================
Measure[3]      Pitch  Duration    Type
                -----  --------  ------
   [1]             D5        18  eighth
   [2]             E5         6    16th
   [3]            F15        18  eighth
   [4]            G15         6    16th
   [5]             A5         6    16th
   [6]             B5         6    16th

======================================
Measure[4]      Pitch  Duration    Type
                -----  --------  ------
   [1]            C16        18  eighth
   [2]             D6         6    16th
   [3]             E6        18  eighth
   [4]            F16         6    16th
   [5]            G16         6    16th
   [6]             A6         6    16th

======================================
Measure[5]      Pitch  Duration    Type
                -----  --------  ------
   [1]            G16        18  eighth
   [2]            F16         6    16th
   [3]             E6        18  eighth
   [4]             D6         6    16th
   [5]            C16         6    16th
   [6]             B5         6    16th

======================================
Measure[6]      Pitch  Duration    Type
                -----  --------  ------
   [1]             A5        18  eighth
   [2]            G15         6    16th
   [3]            F15        18  eighth
   [4]             E5         6    16th
   [5]             D5         6    16th
   [6]           C15         6    16th

======================================
Measure[7]      Pitch  Duration    Type
                -----  --------  ------
   [1]             B4        18  eighth
   [2]             A4         6    16th
   [3]            G14        18  eighth
   [4]            F14         6    16th
   [5]             E4         6    16th
   [6]             D4         6    16th

======================================
Measure[8]      Pitch  Duration    Type
                -----  --------  ------
   [1]            C14        18  eighth
   [2]             B3         6    16th
   [3]             A3        18  eighth
   [4]            C14         6    16th
   [5]             B3         6    16th
   [6]             A3         6    16th

[7] >>     Save ../../../XmlFiles/input A major scale-LRS.xml
File saved successfully to ../../../XmlFiles/input A major scale-LRS.xml

[8] >>
```

Program Logic

The program consists of two main modules and two small secondary data structures. They are:

Filename	Description
RhythmAndClues.cs	The mainline
Interpreter.cs	The main logic of the program
DurationDef.cs	We have to map notes (half, quarter, etc.) to different durations depending on whether we're working with Finale or MuseScore. This class holds that information.
MusicXml.cs	A utility class to access data from a MusicXML file.

RhythmAndClues.cs

This mainline is small. It checks the command line for valid syntax (mostly to ensure that there is a parameter given for the user program), and then it calls the Interpreter to process the file.

Note: This module and others may have conditional compilation (#if/#else/#endif). The code inside these blocks are usually old code from a previous version of the program. Until we get the specs for this program nailed down, I've kept the old code, just in case…

Interpreter.cs

The original concept was to have the user modify the C# source, just slightly (perhaps just a single line or two), each time ze wanted to specify a new file to process or change the rhythm pattern, and then recompile the C# program and run it. This was considered simple enough to be feasible.

However, when I started writing the program, I decided that it was easy enough to write a simple interpreter to process a specialized “language” that the user could supply as an external file and thus ze’d never have to modify the C# program. This was done.

A simple language was defined (see page 2).

The user program was processed in two passes. One thing I hate about dynamic languages is that too often you get a “variable or function undefined” error at runtime (possibly even several hours into the running of the program) when it could have been detected at compile time. I also dislike systems in which a syntax error is found and everything stops until you fix it. Then you have to go through the whole process again just to find the next typo. And again and again…

Parsing was simple (and simplistic). A line with a % in it was a comment for the rest of the line. Blank lines were ignored. Otherwise each line was split into tokens delimited by white space (blanks and/or tabs). The first token was the command and the rest were its arguments.

A simple jump table (see method *SetupCommands*) was used to hold the name of each supported command and the addresses of two methods. One was used to do command-specific syntax checking during pass 1, while the second pointed to the actual functional routine for the second pass.

The rest of the routines in the module were either to support calculating the time signature of the updated score (which currently isn’t working due to the lack of a suitably defined definition of how to calculate the signature), or else the small routines every program has.

DurationDef.cs

As mentioned above, a small class to hold note durations and names. Note that it also has an *IsDotted* field. We probably don’t need this and it will probably be deleted once the rest of the program is finished.

MusicXml.cs

A small class that holds the information we need to glean from a MusicXML file, and some utility functions to return them.