Automating the Addition of Harp Pedals to Standard Sheet Music

Katie Zucker, Department of Computer Science

College of Engineering Computer Science

Abstract

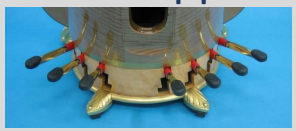
For most instruments, playing a different note (e.g., C natural vs. C sharp vs. C flat) requires moving your finger to a different location on the instrument. C natural is one fret on a guitar; C flat is a fret below the C natural; and C sharp is a fret above. On a piano, playing C flat means playing the key right below C natural, and playing C sharp means playing the key above. Most instruments work this way, with separate notes for each pitch in the musical chromatic scale.

Harpists have a different system for playing sharps or flats: they use pedals to change the intonation of the harp's strings. There is one pedal for each natural note, A, B, C, D, E, F, and G. Each pedal has three positions: natural, sharp, and flat. For instance, if the harpist wants a C flat, he moves the C pedal to the flat position, which makes all the C's on the harp C flats. When the harpist wants to play a C natural again, he moves the C pedal back to its natural position.

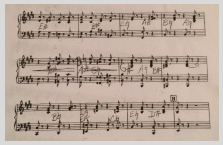
Harpists usually work with sheet music that looks like standard piano music. To account for the pedal changes, they add symbols like "Ab" (which indicates moving the A pedal to the flat position) below the staff directly under the place they want the pedal change (such as the note that requires the pedal change). Whenever a harpist gets a piece of music, the harpist usually needs to spend this time reading through the music, figuring out what the pedals need to be to account for any accidentals in the music. These notations are written in as "pedals" under the staff.

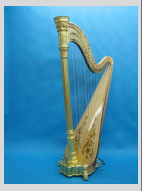
This process of "adding pedals" takes time, and it is also prone to human error. Because the process is objective and algorithmic, I reasoned that a computer can do it. My program, "Harp Pedal Writer," implemented as a plugin in MuseScore2, takes an XML sheet music file as input, adds pedal markings to it, and formats it as another XML file as output. To use it, one must first install the plugin "Harp Pedal Writer" in the free, open-source sheet music writing tool MuseScore2, which is similar to other music writing tools such as Sibelius or Finale. Once the user has written music in MuseScore2 (music written in MuseScore2 is saved as an XML file), the user can click on "Add Pedals" from "Harp Pedal Writer" in the "Plugins" menu. This will automatically add harp pedal markings to the opened sheet music. The plugin makes use of MuseScore2's tools of text boxes to add the pedals, so the new file is simply the old file plus a text box for each pedal. This new file can then be saved as an XML, creating sheet music that is now harpist-friendly and ready for use. I tested my project by converting progressively complex traditional sheet music to include the pedal notation through the automation provided by my software.

What are harp pedals?

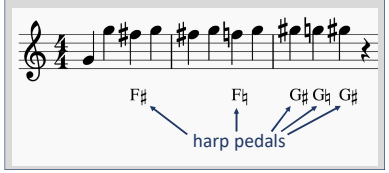


Harps have 7 pedals near the base, and each pedal changes one note on a scale: A, B, C, D, E, F, and G. Each pedal has 3 positions: natural, flat, and sharp. When harpists play F#, for instance, they have to change their F pedal to the sharp position. When using standard sheet music, harpists add harp pedal notation below the staff to indicate when harp pedals should be changed and which harp pedals to change to account for the accidentals.

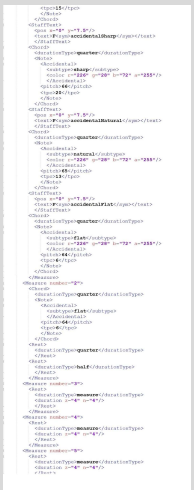




My program, "Harp Pedal Writer," analyzes an existing piece of music and automates the generation of harp pedals into the music score.



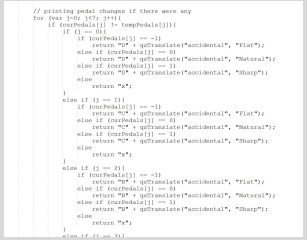
XML Sheet Music



MuseScore2

MuseScore2 is a free, open-source sheet music editor. It works like the more well-known sheet music editors Sibelius and Finale, but all the code is open source, which allows modification of sheet music in MuseScore2 by changing its XML code. The open source model lends itself excellently to the creation of plugins, or bits of code that modify the functionality of MuseScore2 itself so that modified versions of sheet music can be created. My project was the creation of a new plugin, "Harp Pedal Writer," that identifies and notates pedal changes for harp music.

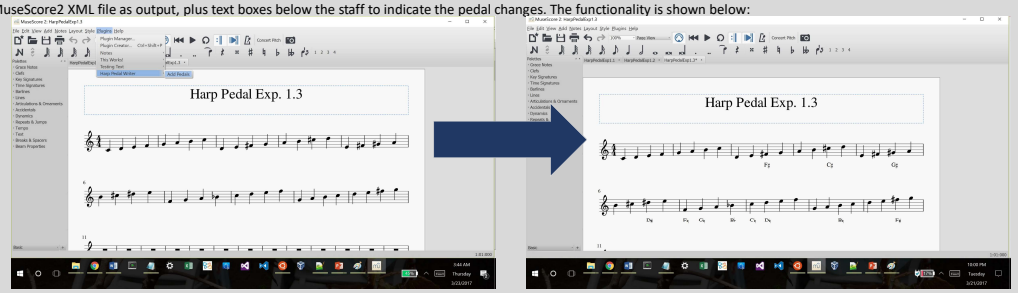
QML plugin code



```
// printing pedal changes if there were any
for (var j=0; j<7; j++){
    if (curPedals[j] != tempPedals[j]){
        if (j == 0){
            if (curPedals[5] == -1)
                return "D" + qsTranslate("accidental", "Flat");
            else if (curPedals[j] == 0)
                return "D" + qsTranslate("accidental", "Natural");
            else if (curPedals[j] == 1)
                return "D" + qsTranslate("accidental", "Sharp");
            else
                return "x";
        }
        else if (j == 1){
            if (curPedals[6] == -1)
                return "E" + qsTranslate("accidental", "Flat");
            else if (curPedals[j] == 0)
                return "E" + qsTranslate("accidental", "Natural");
            else if (curPedals[j] == 1)
                return "E" + qsTranslate("accidental", "Sharp");
            else
                return "x";
        }
        else if (j == 2){
            if (curPedals[0] == -1)
                return "F" + qsTranslate("accidental", "Flat");
            else if (curPedals[j] == 0)
                return "F" + qsTranslate("accidental", "Natural");
            else if (curPedals[j] == 1)
                return "F" + qsTranslate("accidental", "Sharp");
            else
                return "x";
        }
        else if (j == 3){
```

Implementation and Results

"Harp Pedal Writer" steps through sheet music, keeping track of pedals and printing pedal changes when necessary. It takes a MuseScore2 XML file as output, plus text boxes below the staff to indicate the pedal changes. The functionality is shown below:

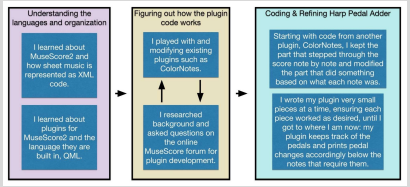


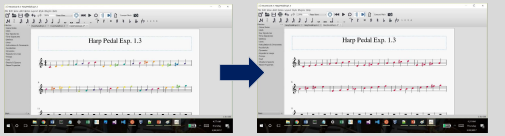
My plugin uses QML code, like other plugins for MuseScore2, to modify the XML sheet music output. I used code and ideas from several other plugins, especially ColorNotes and NoteNames, used the code for MuseScore2 and its documentation, and I used the MuseScore forum to learn about writing plugins and to figure out how to implement my program.

For the implementation of the harp pedal adding functionality, I used two arrays to keep track of the pedals: curPedals, which keeps track of the positions the pedals are currently in as it steps through the score, and keyPedals, which keeps track of what positions the pedals would be in if they matched the key signature. Since there are seven pedals, the arrays are indexed 0-6, corresponding to D, C, B, E, F, G, and A to match the order of the pedals on a harp left to right. The arrays hold integers: -1 to indicate flats, 0 to indicate naturals, and 1 to indicate sharps. There is a function which steps through the score, each measure, and each chord, which is nearly the identical function that steps through the music that I copied from the ColorNotes plugin. There is another function that is called by the first function, checkPedals, that implements my algorithm. In this function, I first set another array tempPedals to remember what my curPedals array was initially. Then, I check for accidental changes an set curPedals accordingly, and I finally check any differences between curPedals and tempPedals, and if there are differences, I print the pedal changes.

Implementation Process

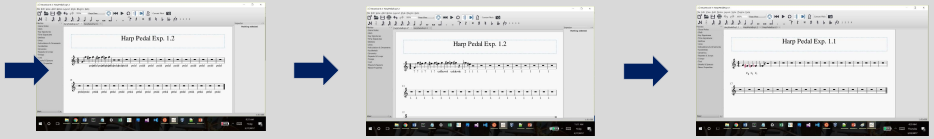
For implementation, I repeatedly tested small additions of code to observe how the XML changed. I experimented with adding text, keeping track of pedals, and finding accidentals. My main steps are shown in the flowchart, and pictures of different stages of development are included.





Observing how the ColorNotes plugin worked

Modifying the Colornotes plugin to make all the notes red



Learning how to add text

Figuring out different types of accidentals

Adding and remembering first pedals

Does new accidentals but doesn't remember keyPedals

Successful output

Future Steps

Right now, "Harp Pedal Writer" determines which pedals need to be changed and adds text boxes directly under the note for each pedal change. The core functionality of adding the harp pedals is complete, but there are a few things I would like to add to make this program truly match a human's ability to add harp pedals to sheet music.

I would like to add functionality so that it spaces out these pedal changes: instead of simply writing the pedals overtop of each other if there are multiple changes in the same chord, the pedal changes should be spaced earlier if possible. Harpists also need a certain amount of time to move their feet from previous pedal changes to new pedal changes, so there is a certain desired amount of spacing between pedal changes. I would eventually like to add conditions in my code so that the plugin will automatically space out the pedal changes, or, if it is impossible, color the notes red. "Harp Pedal Writer" should also be able to add pedals for two clefs, since harpists play in both the F and the G clefs (harp music looks like piano music), and it should also allow two pedal changes, written one above the other, if one pedal is on the left side and the other pedal on the right side of the harp, since harpists can use both feet.

Finally, importantly, once I finish these additions, I am going to add my plugin to the database of MuseScore2 plugins so that everyone can use it. Eventually, I hope to expand this and connect it to another open-source software called Audiveris, so that harpists could scan sheet music into MuseScore2, then use my plugin to add harp pedals. In this manner, harpists could easily add harp pedals to all their music.