Audiveris Handbook

This documentation applies to release 5.9 and later.

TABLE OF CONTENTS

Intended audience

This handbook is meant for the Audiveris end-user. To ease the reading for a newcomer as for a more advanced user, it is organized as a progressive sequence of chapters.

It is just a user manual; a true developer documentation is still to be written. Some material is already made available, through the Audiveris Wiki and through the description of the `.omr` file format, to ease the software learning curve for any potential developer.

Overview

When questioned about Audiveris in December 2023, ChatGPT answered:

> Audiveris is an open-source optical music recognition (OMR) software. OMR technology is designed to recognize printed music notation from scanned images or photos and convert it into a digital format that can be edited and played back. Audiveris specifically focuses on converting scanned sheet music into MusicXML format.

> The software is developed in Java and is available for various operating systems, including Windows, macOS, and Linux. It can be used to digitize printed sheet music, making it easier to edit, share, and playback on digital devices.

Not that bad at all! But let’s try to describe it on our own…

Audiveris is an open source software, published under the AGPL V3 license.

It is an **O**ptical **M**usic **R**ecognition (OMR) application, featuring an OMR engine coupled with an interactive editor.

It recognizes music from digitized images and converts it into computer-readable symbolic format. This enables further music processing by any external notation editor, notably: digital playback, transposition and arrangement.

Audiveris provides outputs in two main digital formats: its own OMR format and the standard MusicXML format.

- OMR is the Audiveris specific format, a documented open source format based on XML.
- MusicXML is a *de facto* standard. It has been designed for score interchange and is today supported as input/output by almost every music notation program.

Not every kind of sheet music can be handled. Audiveris engine has been designed to process scores written in the *Common Western Music Notation* (CWMN) with the following limitations:

- Handwritten scores aren’t supported (only printed scores are),
- Only common musical symbols are supported.

Because the accuracy of the OMR engine is still far from perfection, the Audiveris application provides a graphical user interface specifically focused on quick verification and manual correction of the OMR outputs.

External sophisticated music editors, such as MuseScore or Finale, can be used on the Audiveris MusicXML output. They can even be directly connected via simple plugins to ease the information handover from Audiveris to these external editors.

Handbook structure

This documentation has been restructured for the 5.4 release, according to the principles of the DIVIO Documentation System, around four different functions:

- Tutorials – To get started
  - Installation of Audiveris application
  - Quick tour from a score capture image to its playback by a music

    sequencer
  - Introduction to the main concepts covered by the OMR process
  - Presentation of the graphical user interface
- How-to guides – For precise tasks
  - Frequent tasks like setting book parameters and driving the OMR pipeline
  - Operating in batch through the command line interface
  - Using the graphical interface to edit the OMR results
  - Processing of specific musical items
  - Advanced tasks like sampling symbols and training the neural glyph classifier
- Reference – Comprehensive technical descriptions
  - Menus, boards, folders, outputs, editors
  - Known limitations, history of updates
- Explanation – How it works
  - Steps internals

Tutorials

This part of the handbook is essential for the new comer.

It is focused on how to make Audiveris up and running, and to lead the user by the hand in discovering the application – a quick tour, the main concepts, the main window.

TABLE OF CONTENTS

Installation

Using installers

For **Windows**, **macOS** and **Linux**, Audiveris provides installers available in 64-bit version.

An installer simplifies installation but has the main disadvantage of being limited to the latest official version, which may be several months old.

Building from sources

For **Windows**, **macOS**, **Linux** and **ArchLinux**, you have the ability to **clone and build** from the source material.

The main advantage is to benefit from all the bug fixes and improvements that are continuously published on the Audiveris project site in its “development” branch.

TABLE OF CONTENTS

Tutorials / Installation / Installing binaries

Installing binaries UPDATED IN 5.9

TABLE OF CONTENTS

Installers

Since version 5.5, Audiveris provides installers for Windows, for Linux and for macOS.

These installers are based on the same structure:

1. The application comes with its own Java Runtime Environment (JRE). Therefore, there is no need for the user to install a specific JRE.
2. The application comes with *no* pre-installed OCR languages, but offers a runtime dialog box allowing the user to install any desired OCR language(s). The application is then responsible for picking up the right version of the language files on the Tesseract site and for installing them in the user environment.

The installers files can be downloaded from the assets of a recent release available on the Audiveris releases page.

The name of each installer file is formatted as:

```
Audiveris-<version>-<OS>-<Architecture>
```

For example, the assets of the 5.7.0 release contain these files:

File name	Size	Role	Option
Audiveris-5.7.0-macosx-arm64.dmg	66.5 MB	macOS installer	
Audiveris-5.7.0-macosx-x86_64.dmg	68.2 MB	macOS installer	
Audiveris-5.7.0-ubuntu22.04-x86_64.deb	62.7 MB	Linux installer	22.04
Audiveris-5.7.0-ubuntu24.04-x86_64.deb	62.5 MB	Linux installer	24.04
Audiveris-5.7.0-windows-x86_64.msi	66.3 MB	Windows installer	
Audiveris-5.7.0-windowsConsole-x86_64.msi	66.3 MB	Windows installer	Console
Audiveris_Handbook-5.7.0.pdf	12.9 MB	PDF manual	
Audiveris_Schemas_Doc-5.7.0.pdf	1 MB	Developer	

File name	Size	Role	Option
		doc	

Remarks:

- for **macOS**, two different installers are provided:
  - One for the `arm64` architecture
  - One for the `x86_64` architecture
- for **Linux**, installers are provided for two Ubuntu versions:
  - Old `22.04`
  - New `24.04`
- for **Windows**, two installers are provided:
  - One without console, which displays just the graphic user interface. This is the recommended variant.
  - One with console, where a Terminal window is launched together with the graphic user interface. This variant can be useful to display error messages.
- The **PDF** version of the Audiveris **handbook** can be useful if you don't have a permanent Internet access, since its content is identical to the online version.
- The **Schemas_Doc** is meant for the developers. It documents the entities `Book`, `Sheet` and `PlayList` which compose the Audiveris `.omr` project files.

The downloaded installer file will be used to install the application in the target OS, as detailed in the following sections.

> **NOTE**
> Once the application is launched, OCR languages can be downloaded directly from within the Audiveris application. See the OCR languages section.

Beside these installers, Audiveris used to provide an additional installer – actually a **Linux/Flatpak** package – also with a suitable JRE included. This package can be installed directly from the Flathub site.
However, the future of this Flatpak package is not clear, for lack of skill and/or manpower.

Windows community repository

NEW IN 5.9

Thanks to Matthew Watkins suggestion, Audiveris is now available on the Windows community repository.

Anybody can try this command in a terminal window:

> winget show Audiveris

which gives (as of this writing):

```
Version: 5.8.1
Publisher: audiveris.org
Description: Optical Music Recognition
License: AGPLv3
Installer:
  Installer Type: wix
  Installer Locale: en-US
  Installer Url: https://github.com/Audiveris/audiveris/releases/download/5.8.1/Audiveris-5.8.1-win
-x86_64.msi
  Installer SHA256: 0bafcdf120a7c4bece0edc95f295611493c4550a7fa3b67108f880ec25b18716
  Offline Distribution Supported: true
```

Installation

In a terminal window, enter this command:

> winget install Audiveris

which gives (after being prompted for confirmation by the User Account Control):

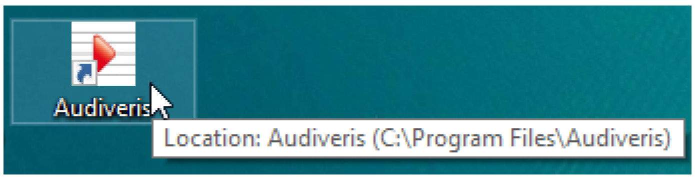
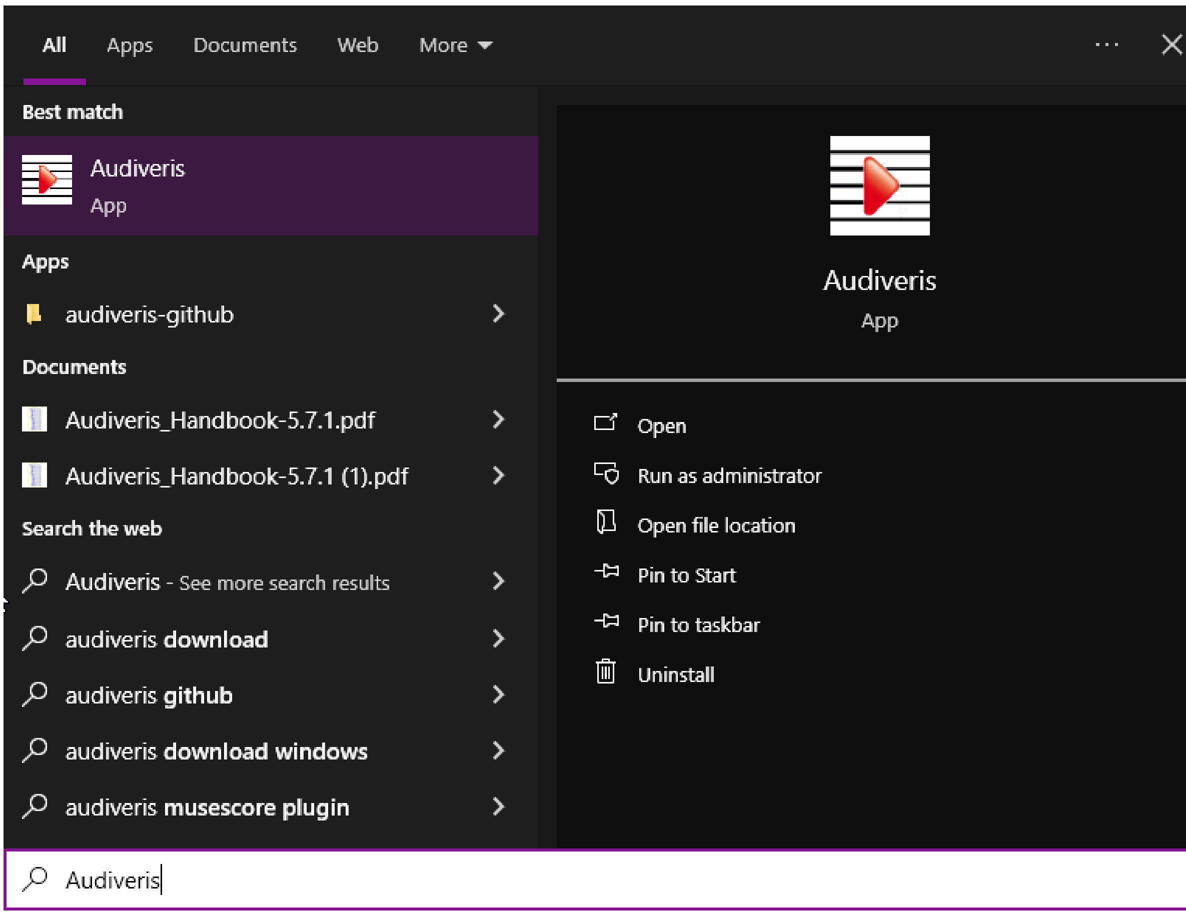
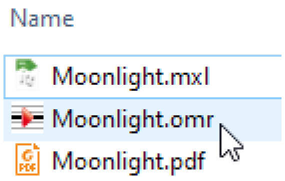
```
Found Audiveris [audiveris.org.Audiveris] Version 5.8.1
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Downloading https://github.com/Audiveris/audiveris/releases/download/5.8.1/Audiveris-5.8.1-windows-
  ██████████████████████████████  68.0 MB / 68.0 MB
Successfully verified installer hash
Starting package install...
Successfully installed
```

Note: The WinGet command references the Audiveris Windows installation file available in the release resources on GitHub. However, running WinGet does not raise any additional questions regarding unrecognized app, the license agreement, the installation folder, or shortcuts.

There is no shortcut in the start menu, just a shortcut on the desktop.

Running

The application can be launched in different ways:

Way	Illustration
Double-clicking on the Audiveris icon located on Windows **desktop**	 Audiveris Location: Audiveris (C:\Program Files\Audiveris)
Entering Audiveris in the Windows **search area** and selecting an action	 All Apps Documents Web More Best match Audiveris App Apps audiveris-github Documents Audiveris_Handbook-5.7.1.pdf Audiveris_Handbook-5.7.1 (1).pdf Search the web Audiveris - See more search results audiveris download audiveris github audiveris download windows audiveris musescore plugin Audiveris App Open Run as administrator Open file location Pin to Start Pin to taskbar Uninstall Audiveris
Double-clicking on a `.omr` **file**, since the `.omr` file extension (which represents a	

Way	Illustration
Book) has been associated with Audiveris application	
In a terminal window, entering a **command** refering to the program location	“C:\Program Files\Audiveris\Audiveris.exe” `<potential arguments>`

Again, since the application is installed by WinGet, it will not generate any alerts from the antivirus.

Uninstallation

In a terminal window, enter this command:

> winget uninstall Audiveris

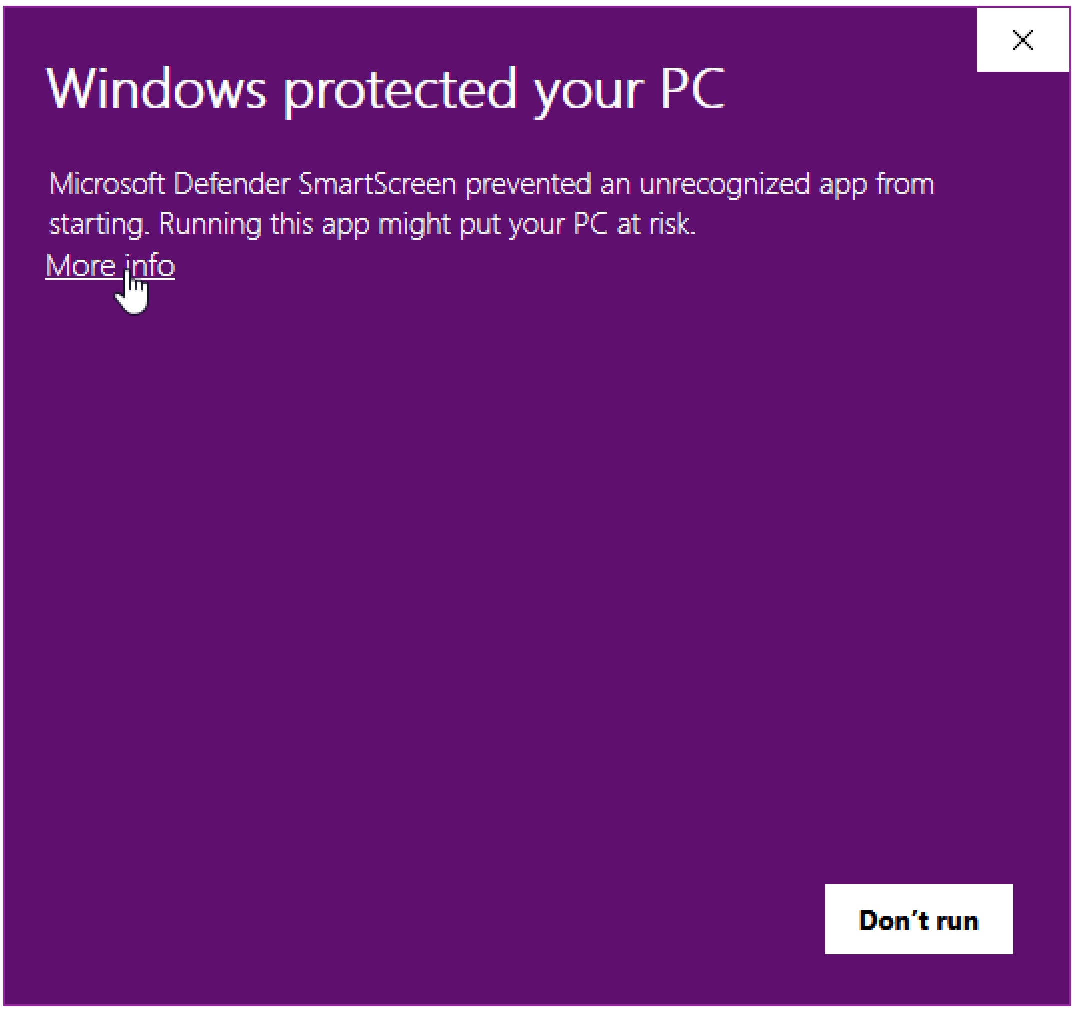
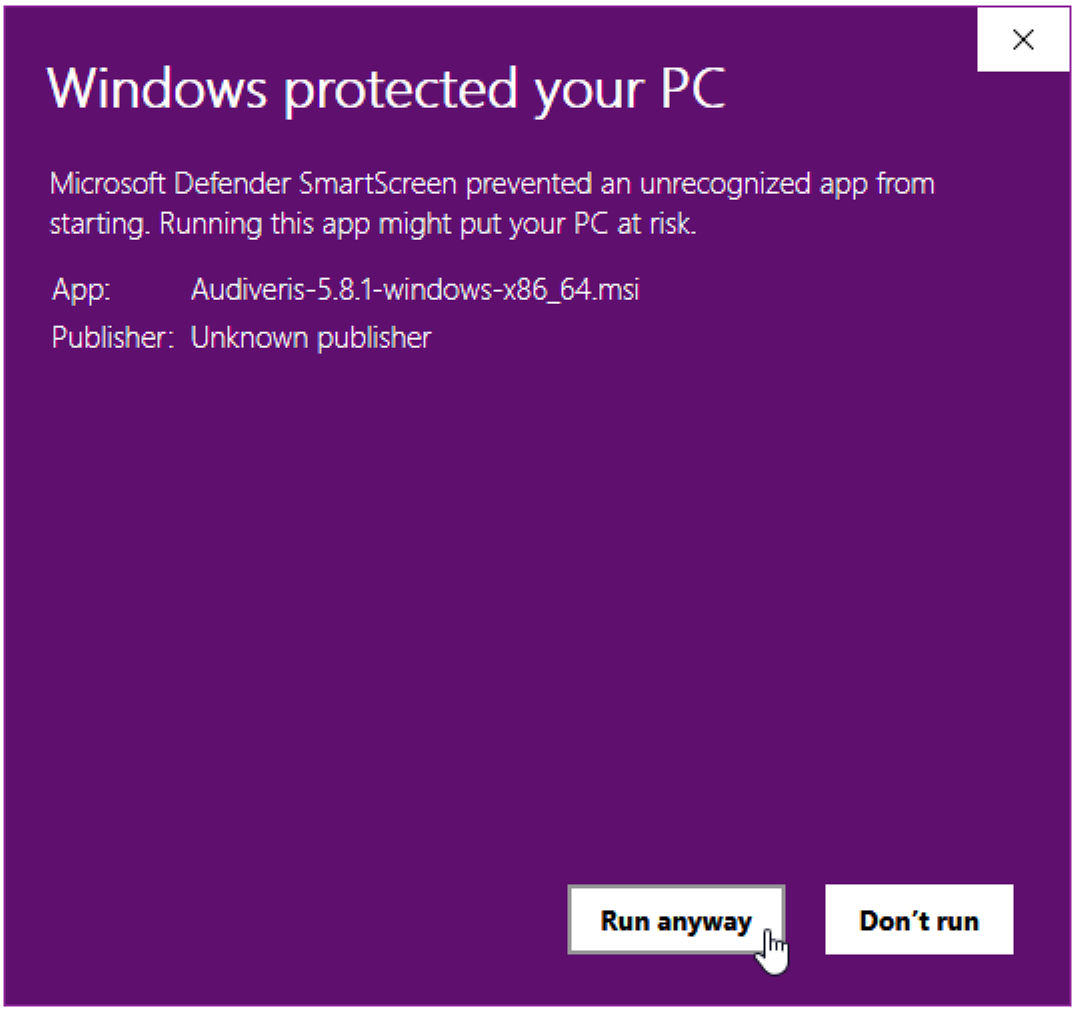
This will give (after being prompted for confirmation by the User Account Control):

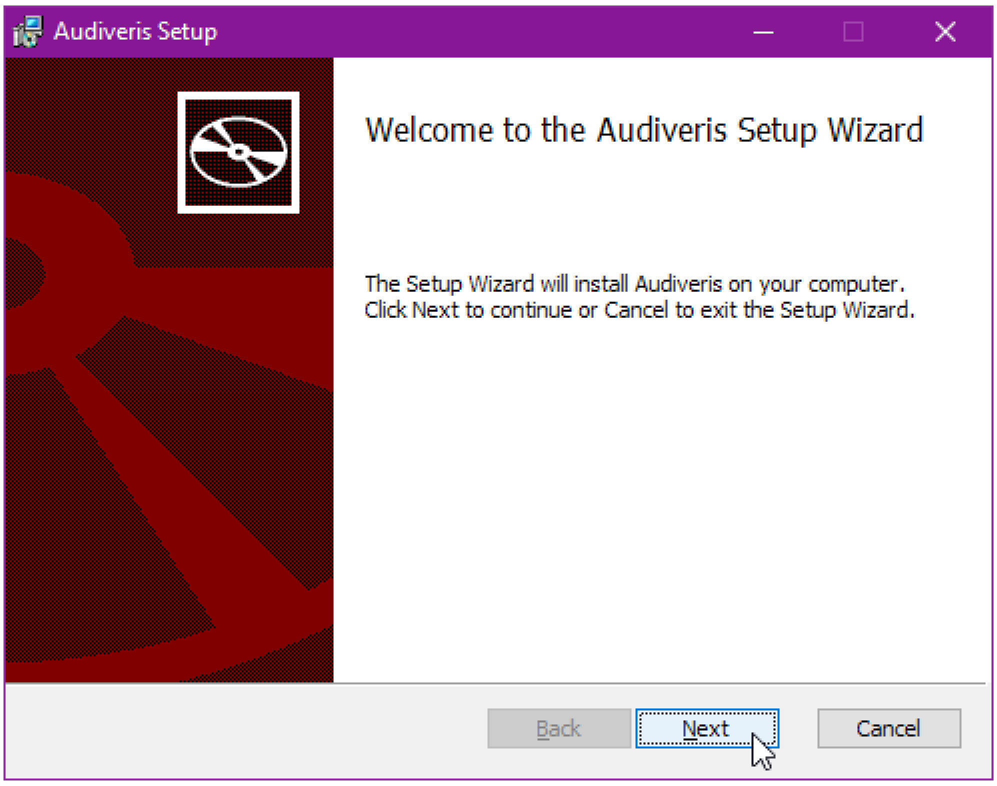
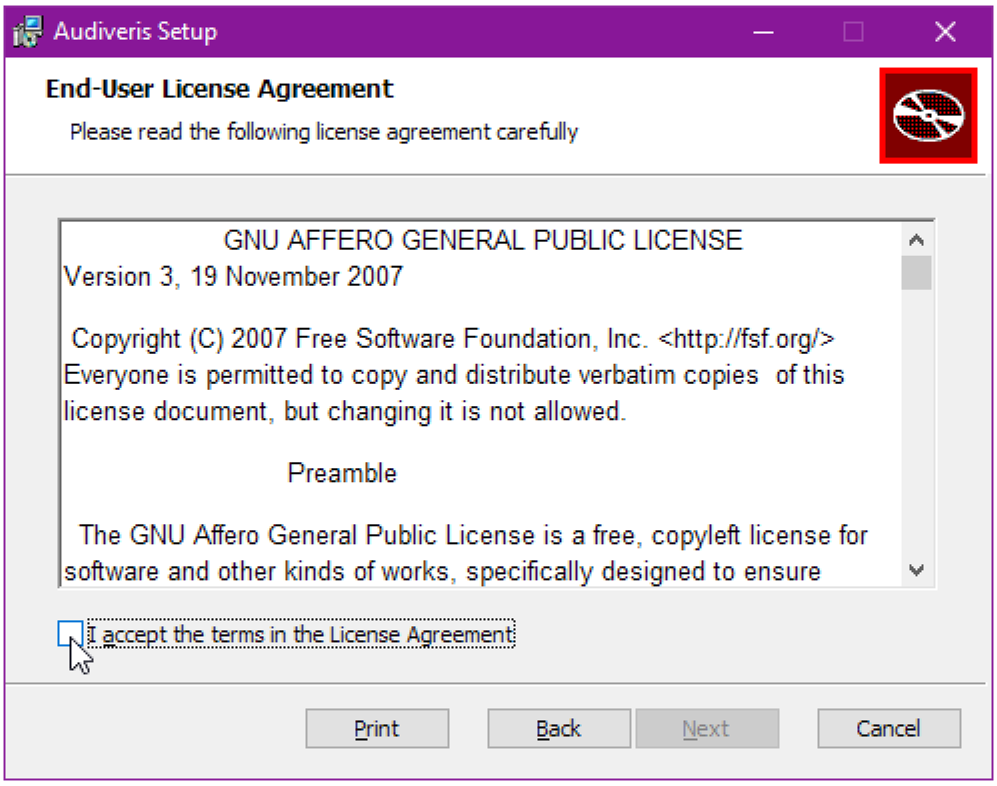
```
Found Audiveris [audiveris.org.Audiveris]
Starting package uninstall...
Successfully uninstalled
```

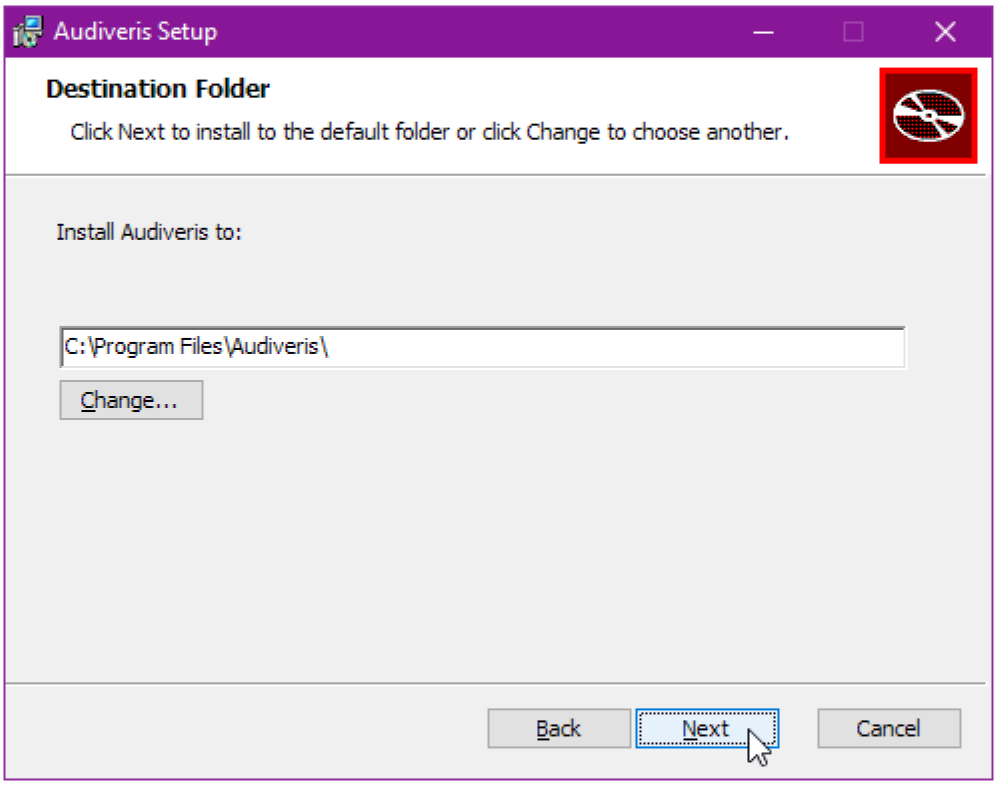
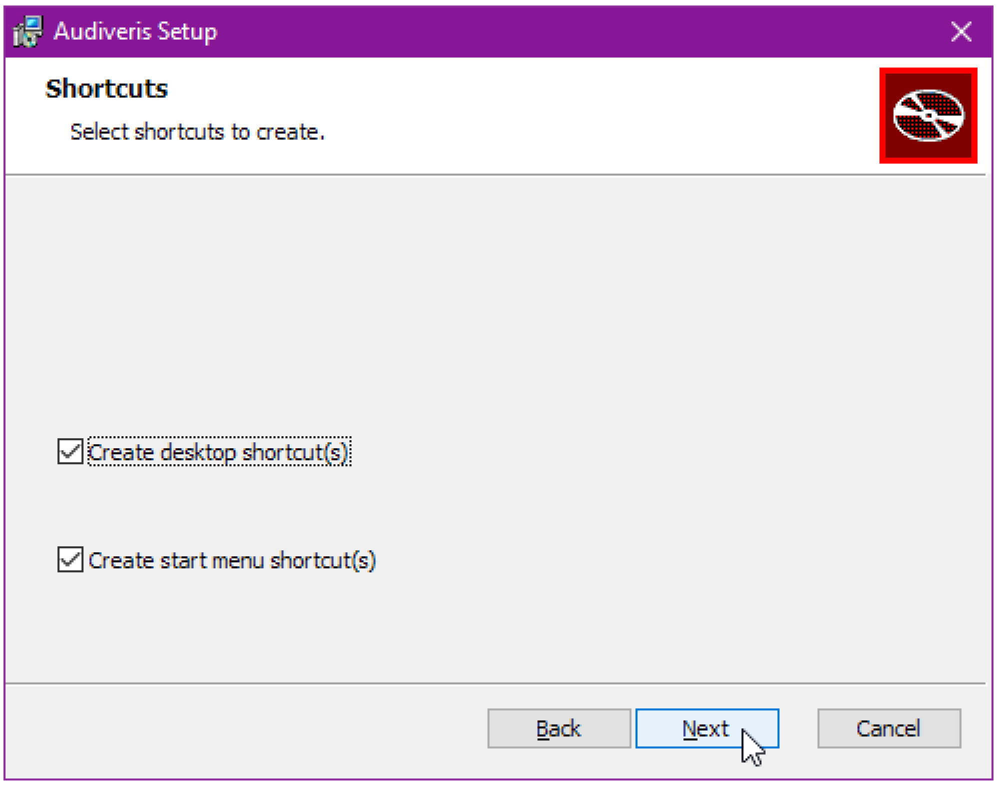
Windows installer

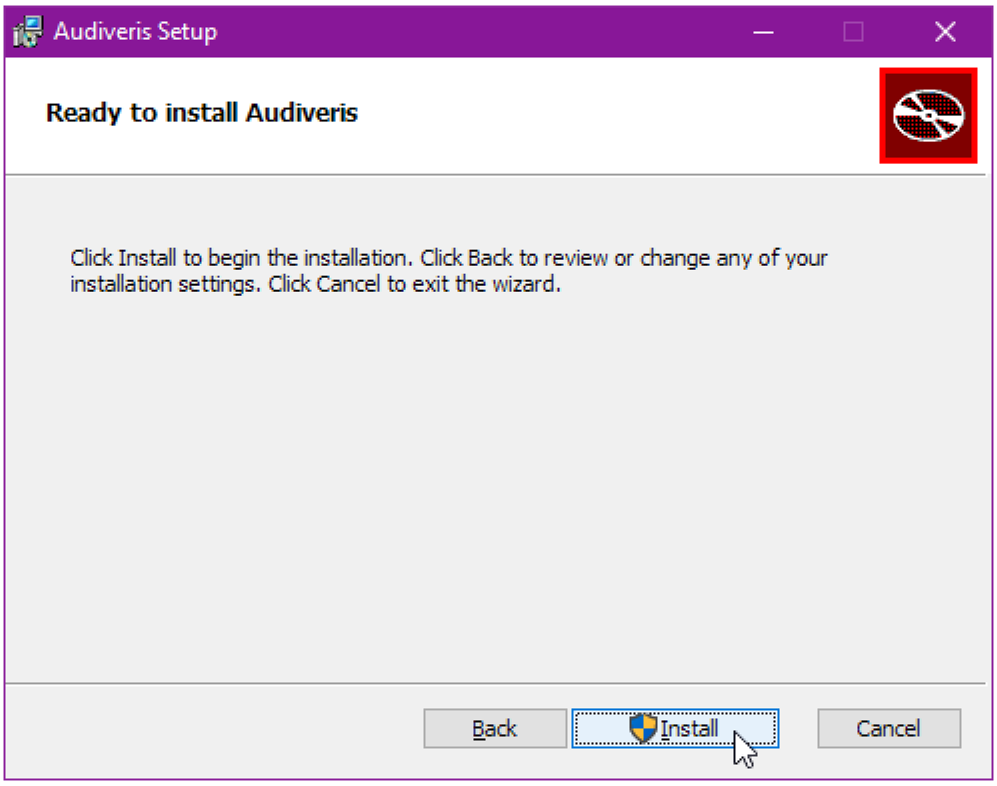
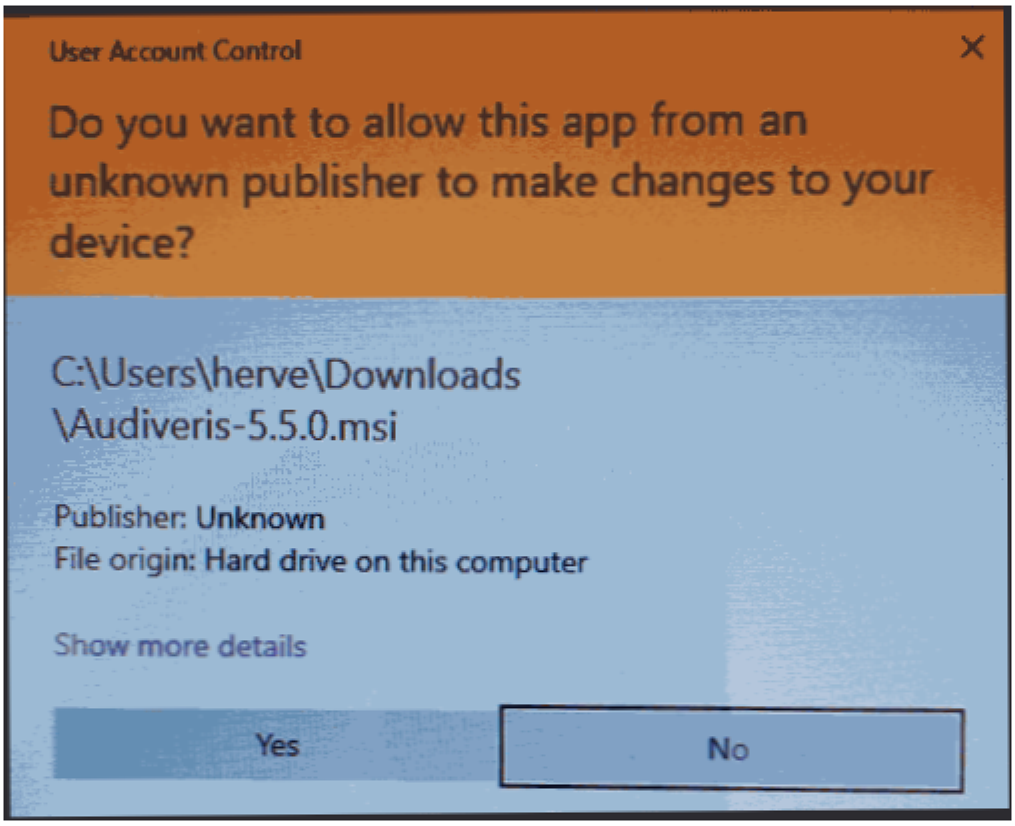
Installation

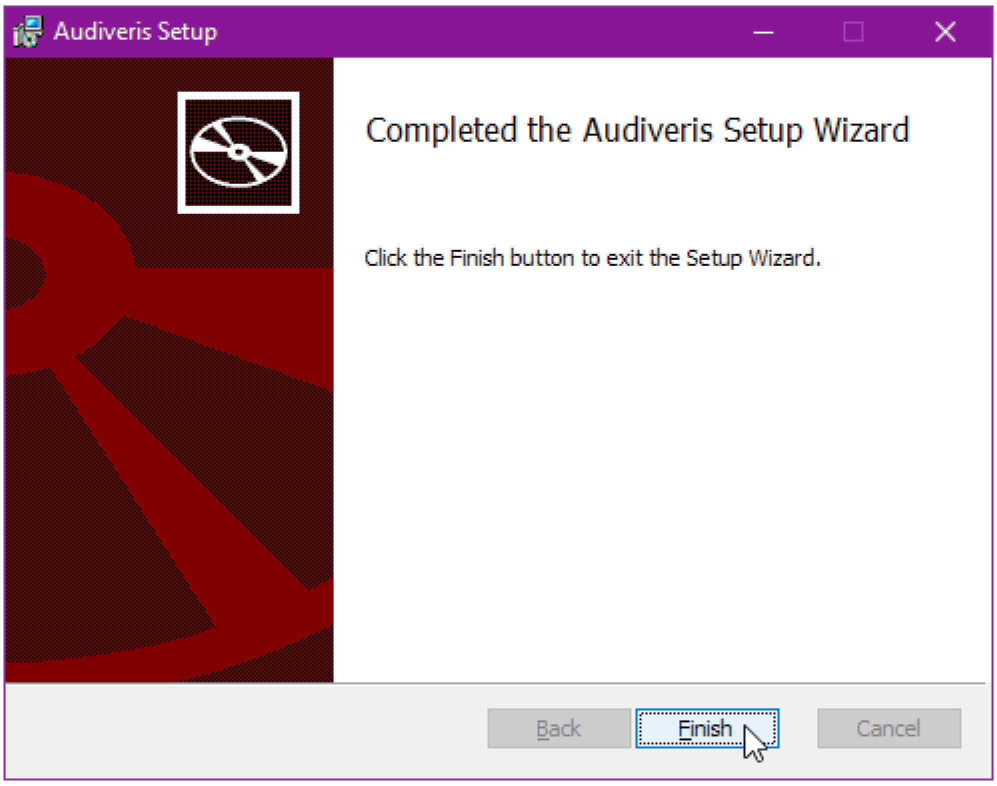
Double-click the `Audiveris-<version>-windows-x86_64.msi` file in your `Downloads` folder (or wherever it’s saved).

Action	Dialog
The double-click may have raised an alert from Microsoft Defender. Click on “More info” (or press “Don’t run” to abandon)	 Windows protected your PC Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.
Press “Run anyway” to keep going (or press “Don’t run” to abandon)	 Windows protected your PC Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk. App: Audiveris-5.8.1-windows-x86_64.msi Publisher: Unknown publisher

Action	Dialog
Now you get the `Audiveris Setup` dialog box	 Welcome to the Audiveris Setup Wizard The Setup Wizard will install Audiveris on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
The license agreement is displayed. If you agree, tick the checkbox and then click on `Next`	 End-User License Agreement Please read the following license agreement carefully GNU AFFERO GENERAL PUBLIC LICENSE Version 3, 19 November 2007 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure ☐ I accept the terms in the License Agreement

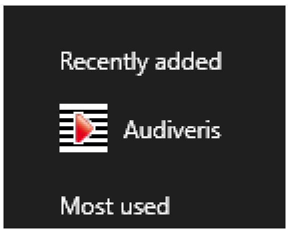
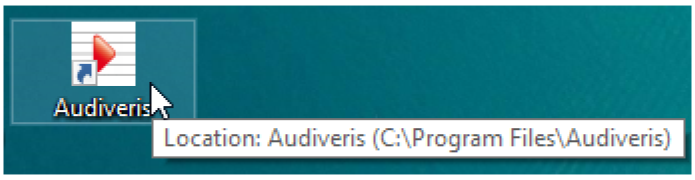
Action	Dialog
The default installation folder can be changed	 Audiveris Setup **Destination Folder** Click Next to install to the default folder or click Change to choose another. Install Audiveris to: C:\Program Files\Audiveris\
By default, the installer creates one shortcut on the desktop and one shortcut in the Windows start menu	 Audiveris Setup **Shortcuts** Select shortcuts to create. ☑ Create desktop shortcut(s) ☑ Create start menu shortcut(s)

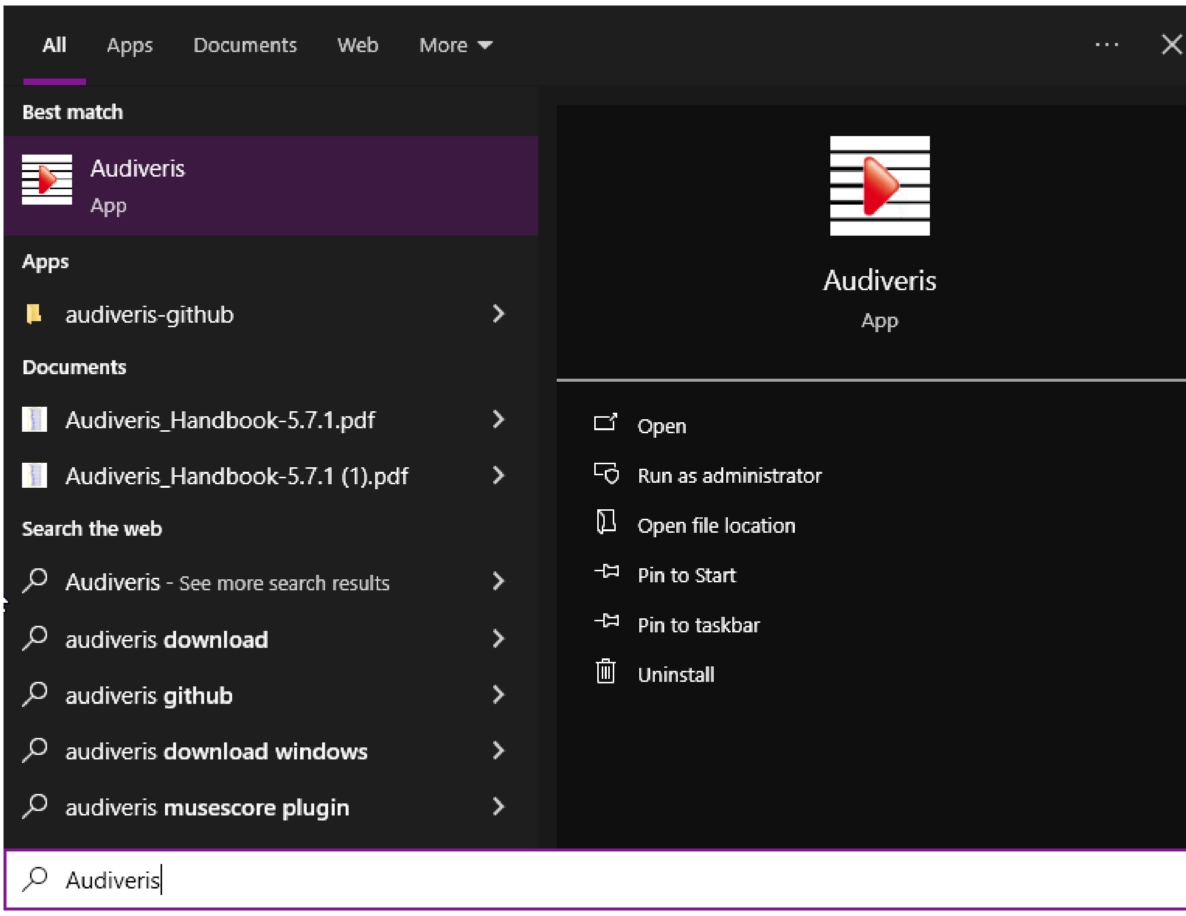
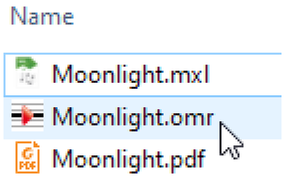
Action	Dialog
The installation is now launched	 Audiveris Setup **Ready to install Audiveris** Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
You may get an alert from Windows user account control, since Audiveris is not from a “known” publisher. Click on `Yes` to allow.	 User Account Control Do you want to allow this app from an unknown publisher to make changes to your device? C:\Users\herve\Downloads\Audiveris-5.5.0.msi Publisher: Unknown File origin: Hard drive on this computer

Action	Dialog
The installation is finished	 Completed the Audiveris Setup Wizard Click the Finish button to exit the Setup Wizard.

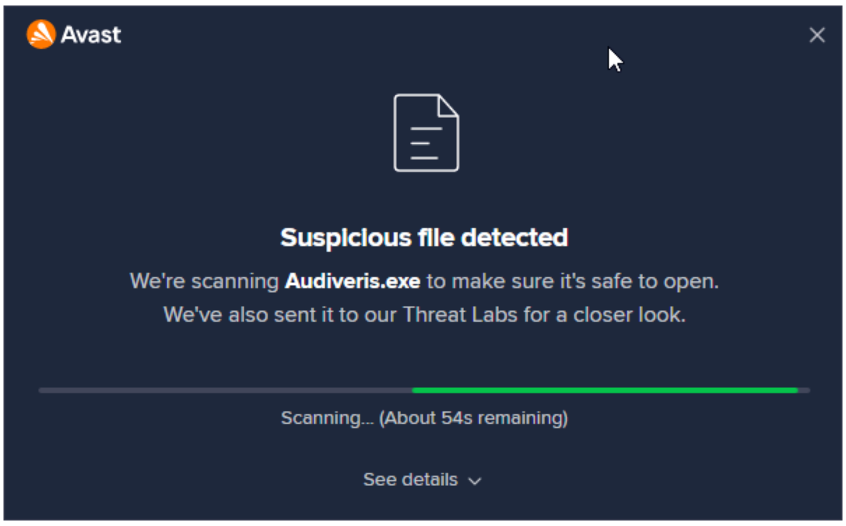
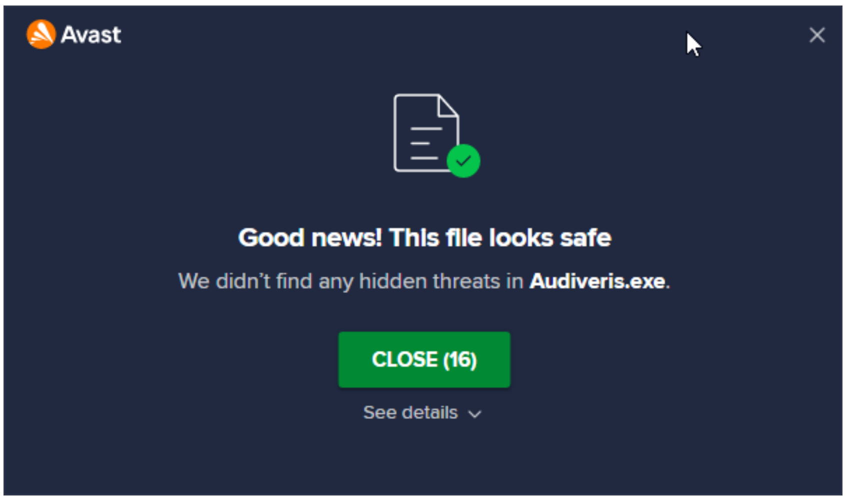
Running

The application can be launched in different ways:

Way	Illustration
Opening the Windows **start menu** and selecting the Audiveris item [1]	
Double-clicking on the Audiveris icon located on Windows **desktop**	 Location: Audiveris (C:\Program Files\Audiveris)

Way	Illustration
Entering Audiveris in the Windows **search area** and selecting an action	
Double-clicking on a `.omr` **file**, since the `.omr` file extension (which represents a Book) has been associated with Audiveris application	
In a terminal window, entering a **command** refering to the program location	“C:\Program Files\Audiveris\Audiveris.exe” `<potential arguments>`

The very first time Audiveris is launched, the anti-virus software may get in the way:

Event	Illustration
The anti-virus has detected a suspicious file	 Avast **Suspicious file detected** We're scanning **Audiveris.exe** to make sure it's safe to open. We've also sent it to our Threat Labs for a closer look. Scanning... (About 54s remaining)
The file looks safe, end of the check.	 Avast **Good news! This file looks safe** We didn't find any hidden threats in **Audiveris.exe**.

Notes

"***Failed to launch JVM***": A couple of users running on Windows 11 reported the same issue 818.

This is reportedly related to something called "Assistive Technology". It is documented in this article https://www.papercut.com/kb/Main/AssistiveTechnologynotfound which provides a solution, copied *verbatim* below.

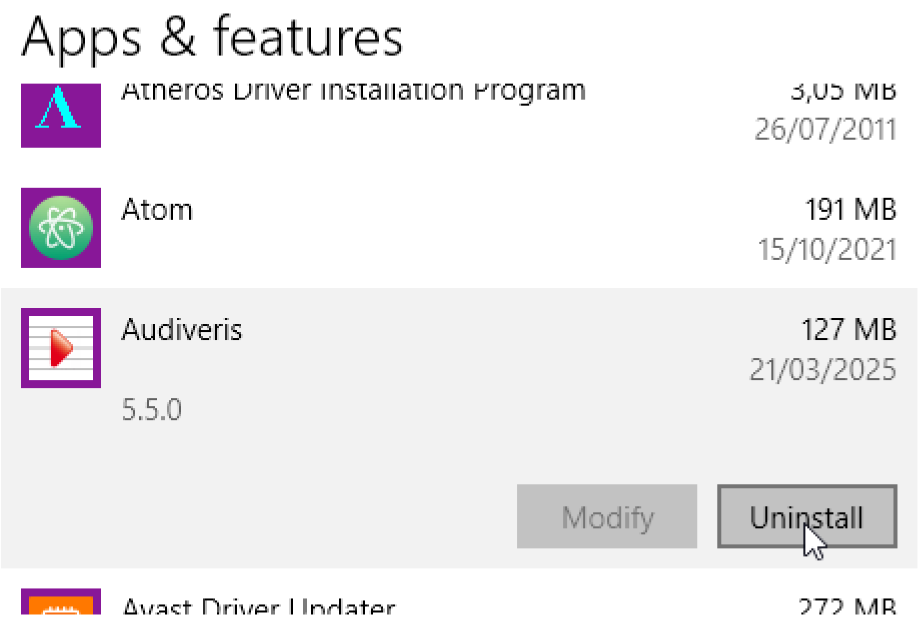
The best fix for this problem is to disable the Access Bridge technology in the Java configuration files. To do this, follow these steps:

1. Navigate to C:\Users<username>
2. Open the file called .accessibility.properties using a standard text editor (Notepad is fine)
3. Find the following lines in the config: ····* assistive_technologies=com.sun.java.accessibility.AccessBridge ····* screen_magnifier_present=true

4. Add a hashmark (#) in front of theses lines
5. Save your changes to the file

Uninstallation

To uninstall the program, open `Windows Settings` (keyboard shortcut is `Windows + I`), and select the `Apps & features` section.

Action	Illustration
In `Apps & features`, look for the Audiveris item and there press the `Uninstall` button	 Apps & features Atheros Driver Installation Program 3,05 MB 26/07/2011 Atom 191 MB 15/10/2021 Audiveris 127 MB 21/03/2025 5.5.0 Avast Driver Updater 272 MB

Linux installer

Installation

Remark: A double-click on the `.deb` installer file would result in the opening of the `App Center` which would choke on audiveris being potentially unsafe, etc.

Instead, in a terminal, use a command like:

```
$ sudo apt install /path/to/Audiveris-<version>-ubuntu<osversion>-x86_64.deb
```

This installs the application in the target folder:

> /opt/audiveris/

This `/opt/audiveris` folder is organized as follows:

```
/opt/audiveris
├── bin
│   └── Audiveris
```

```
├── lib
│   └── ...
└── share
    └── ...
```

Running

In a terminal, use a command like:

```
$ /opt/audiveris/bin/Audiveris <potential arguments>
```

Uninstallation

In a terminal, use one of these commands:

```
# To just uninstall
$ sudo apt remove audiveris

# To remove the application, including configuration files
$ sudo apt purge audiveris
```

Linux/Flatpak installer

Installation

The Audiveris installer for Linux uses the Flatpak utility and is hosted on the standard Flathub repository.

On the Flathub site, you can enter “audiveris” in the search field.
Or you can go directly to the https://flathub.org/apps/org.audiveris.audiveris page.

For a manual install, you can use:

```
$ flatpak install flathub org.audiveris.audiveris
```

Running

To launch the application, run the command:

```
$ flatpak run org.audiveris.audiveris
```

Uninstallation

TODO

macOS installer

This section explains how to install and run the Audiveris application on macOS using the provided DMG installer. Since the installer is not signed with an Apple Developer certificate, you'll need to adjust your macOS privacy settings to allow it to run.

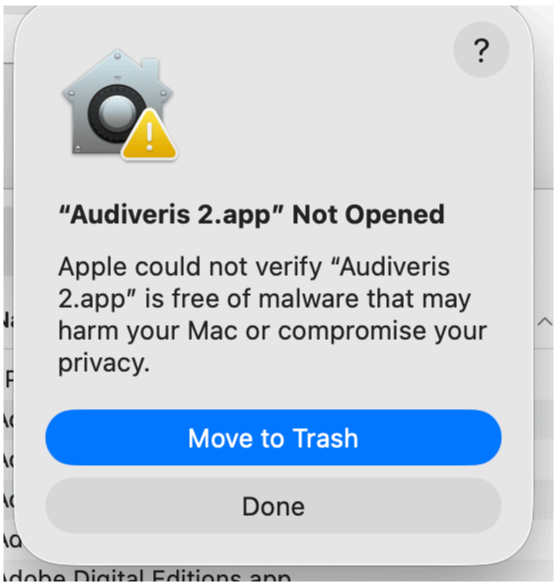
Installation

1. **Obtain the proper DMG File for your architecture**
   - Download or receive the proper `.dmg` file, that is either `Audiveris-<version>-macosx-arm64.dmg` or `Audiveris-<version>-macosx-x86_64.dmg`, from the source (e.g., a contributor or repository release).
2. **Open the DMG**
   - Double-click the chosen `.dmg` file. This mounts the installer as a virtual disk on your desktop or in Finder.
3. **Install the Application**
   - Inside the mounted DMG, you'll see `Audiveris.app`. Drag this file to your **Applications** folder to install it.
   - Once copied, you can eject the DMG by clicking the eject icon next to it in Finder or dragging it to the trash.

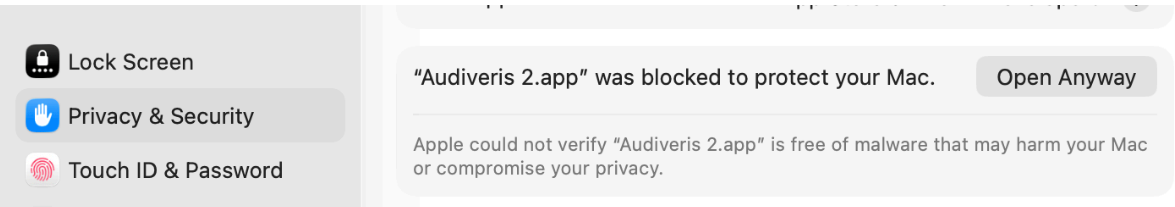
Running

Since the app is not signed, macOS will block it by default. Follow these steps to allow it to run:

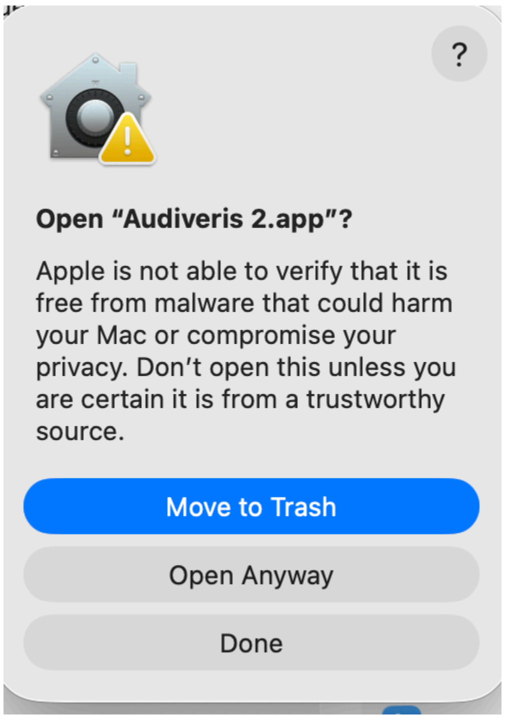
1. **Attempt to Open the App**
   - Go to your **Applications** folder and double-click `Audiveris.app`.
   - You'll likely see a warning: *""Audiveris" cannot be opened because it is from an unidentified developer."*
   - On MacOS Tahoe, you will see a screen as below. Press "Done"

- 
  “Audiveris 2.app” Not Opened

  Apple could not verify “Audiveris 2.app” is free of malware that may harm your Mac or compromise your privacy.

2. **Adjust Privacy Settings**
   - Open **System Preferences** (or **System Settings** on macOS Ventura and later):
     - Click the Apple menu () > **System Preferences** > **Security & Privacy** > **General** tab.
   - At the bottom, you’ll see a message: *“Audiveris” was blocked from use because it is not from an identified developer.*
   - Click **“Open Anyway”** to allow the app to run.
   - 
     “Audiveris 2.app” was blocked to protect your Mac.

     Apple could not verify “Audiveris 2.app” is free of malware that may harm your Mac or compromise your privacy.

3. **Launch the App**
   - Double-click `Audiveris.app` again. You may see one final prompt asking for confirmation—click **“Open”** or **“Open Anyway”**.
   - 
     Open “Audiveris 2.app”?

     Apple is not able to verify that it is free from malware that could harm your Mac or compromise your privacy. Don’t open this unless you are certain it is from a trustworthy source.

- The app should now launch successfully.

Notes

- **Unsigned App**: The lack of a signature is due to the installer not being created with an Apple Developer account. This is a one-time adjustment; once approved, macOS will remember your choice.
- **Troubleshooting**: If the app still won't open, ensure you've completed the privacy settings step. For persistent issues, contact the provider or check the Audiveris documentation.

Uninstallation

TODO

Footnotes

[1] It seems that the Windows installer no longer creates a shortcut in the start menu... ↩

Building from sources (Windows, macOS, Linux, ArchLinux)

NOTE

For GitHub users:

- Audiveris “***master***” branch is updated only when a new release is published.
- Instead, Audiveris development happens continuously in the “***development***” branch, so pull and checkout this *development* branch to get and build the latest version.
- See workflow details in the dedicated Wiki article.

TABLE OF CONTENTS

Tools needed

- Git: version control system.
- Gradle: build tool.
- Java Development Kit (JDK): version 25 (higher numbers may work, to be confirmed).
  This is the minimum Java version needed for the *master* branch. The

  development branch may require a higher version, please check on the Audiveris project site.
  Audiveris 5.9 runs only on 64-bit architectures.

- FreeType library: Unix-like platforms (including macOS) need FreeType in our $PATH to handle those specific PDFs that contain vector graphics.
  Fortunately, every known Unix-like OS distribution already contains a package for FreeType.

- Tesseract OCR: The Tesseract *libraries* are automatically pulled as Gradle dependencies, but we will need some Tesseract *language files* for this OCR to work properly.
  Please check the OCR languages section.

Clone, build and run

To clone the Audiveris project, we can use the following command in a directory of our choice:

```
git clone https://github.com/Audiveris/audiveris.git
```

This will create a sub-directory named “audiveris” in our current directory and populate it with project material (notably source code and build procedure).

Now we move to this “audiveris” project directory:

```
cd audiveris
```

Once in this `audiveris` project directory, we can select the branch we want.
By default, we are on `master` branch.
To use the `development` branch with its latest features, we can use:

```
git checkout development

# And to make sure we grab even the latest updates:
git pull --all
```

We can build the software via the command:

```
# (Linux & Mac, or Cygwin terminal under Windows)
./gradlew build
```

```
# (Windows terminal)
gradlew.bat build
```

We can run the software, with its graphical interface, via the command:

```
# (Linux & Mac, or Cygwin terminal under Windows)
./gradlew run
```

```
# (Windows terminal)
gradlew.bat run
```

Please note that all these commands use the **gradle wrapper** (`gradlew`) which, behind the scenes, takes care of getting and launching the proper gradle version.

Alternative run

The gradle-based run, as described above, makes sure that all our potential modifications are compiled before the application is launched. This is the preferred approach for a developer.

However, if we don’t modify the code and merely want to launch the application we don’t need to go through gradle for each and every run.

Because, once we have built the software with the gradle `build` command as stated above, we now have a `app/build/distributions` subfolder in the project root with tarred/zipped libraries:

```
# This content corresponds to the 5.5.0 version
app/build/distributions/
├── app-5.5.0.tar
└── app-5.5.0.zip
```

We can simply extract either one of these archives into a folder of our choice:

```
# Either extract the .tar archive
tar -xf app/build/distributions/app-5.5.0.tar
```

```
# Or extract the .zip archive
unzip app/build/distributions/app-5.5.0.zip
```

Then, we can repeatedly run Audiveris from those files, appending arguments as desired (`-help` is just used as an example):

```
# (Linux & Mac, or Cygwin terminal under Windows)
./app-5.5.0/bin/Audiveris -help
```

```
# (Windows terminal)
app-5.5.0\bin\Audiveris.bat -help
```

Tutorials / Quick tour

Quick tour

This chapter describes a very early tour of Audiveris, featuring the basic workflow that goes from the *image* of some sheet of music to the *sound* of the music it represents.

Other chapters will cover in more details each step of perhaps more complex workflows, but for the sake of simplicity, let’s assume that we already have:

- Audiveris installed,
- The image of some sheet of music of rather good quality,
- Perhaps a music sequencer installed (MuseScore for example).

All we have to do is, in sequence:

1. Launch Audiveris,
2. Load the input image,
3. Transcribe the image as a score,
4. Export the score in MusicXML,
5. Play the exported MusicXML file with MuseScore.

For the impatient, note that if we have defined a plugin to a sequencer like MuseScore, the sub-sequence of bullets #3, #4 and #5 above could conveniently be replaced by just pressing the plugin button.
This would now give:

1. Launch Audiveris,
2. Load the input image,
3. Press the Plugin button.

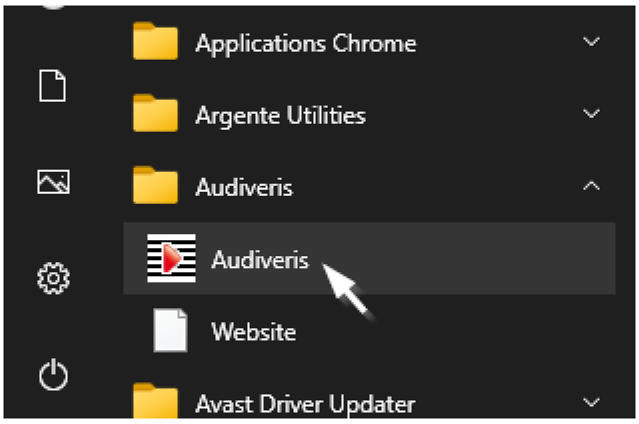
TABLE OF CONTENTS

Launch

For the sake of simplicity, let’s assume that:

1. We are using Windows OS
2. We have installed Audiveris via the provided Windows Installer.
   (This installer can be downloaded from the Audiveris Releases area. See Installing binaries).

We get into the Windows Start Menu, by pressing the Windows logo key on our keyboard or by clicking on the Start icon (left end of the taskbar).
In this menu organized by alphabetical order, we move down to the Audiveris folder, open it, and finally click on the Audiveris executable.



This opens a small terminal window which automatically:

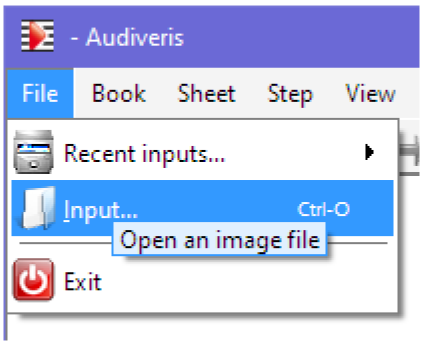
1. Checks our installed Java environment.
   If the check fails, Audiveris is not launched. Instead, a message appears, warning about our Java environment.
2. Launches Java on the Audiveris software.

This results in the display of the Audiveris main window, from which we will be able to operate.

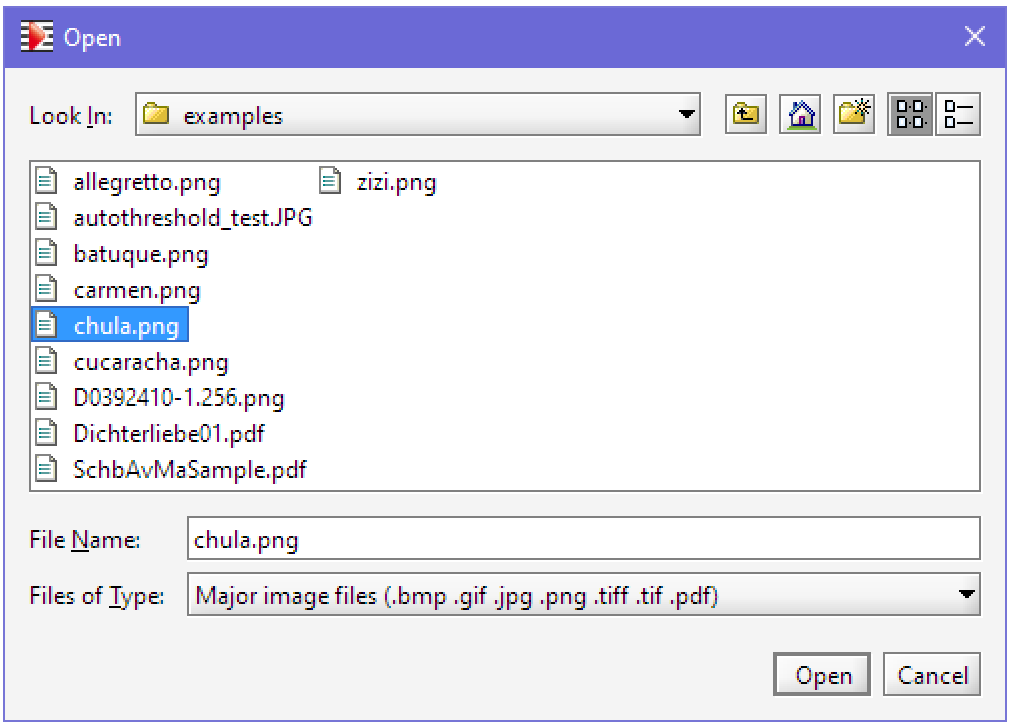
If we don’t use binary installation (either the Windows installer or the Linux flatpak), we should refer to Building from sources to build and then launch the software.

Load

To load an image from disk, we can use the pull-down menu `File → Input`:



An `Open` dialog will show up, allowing us to navigate between folders and finally select an image input file:



Look In: examples

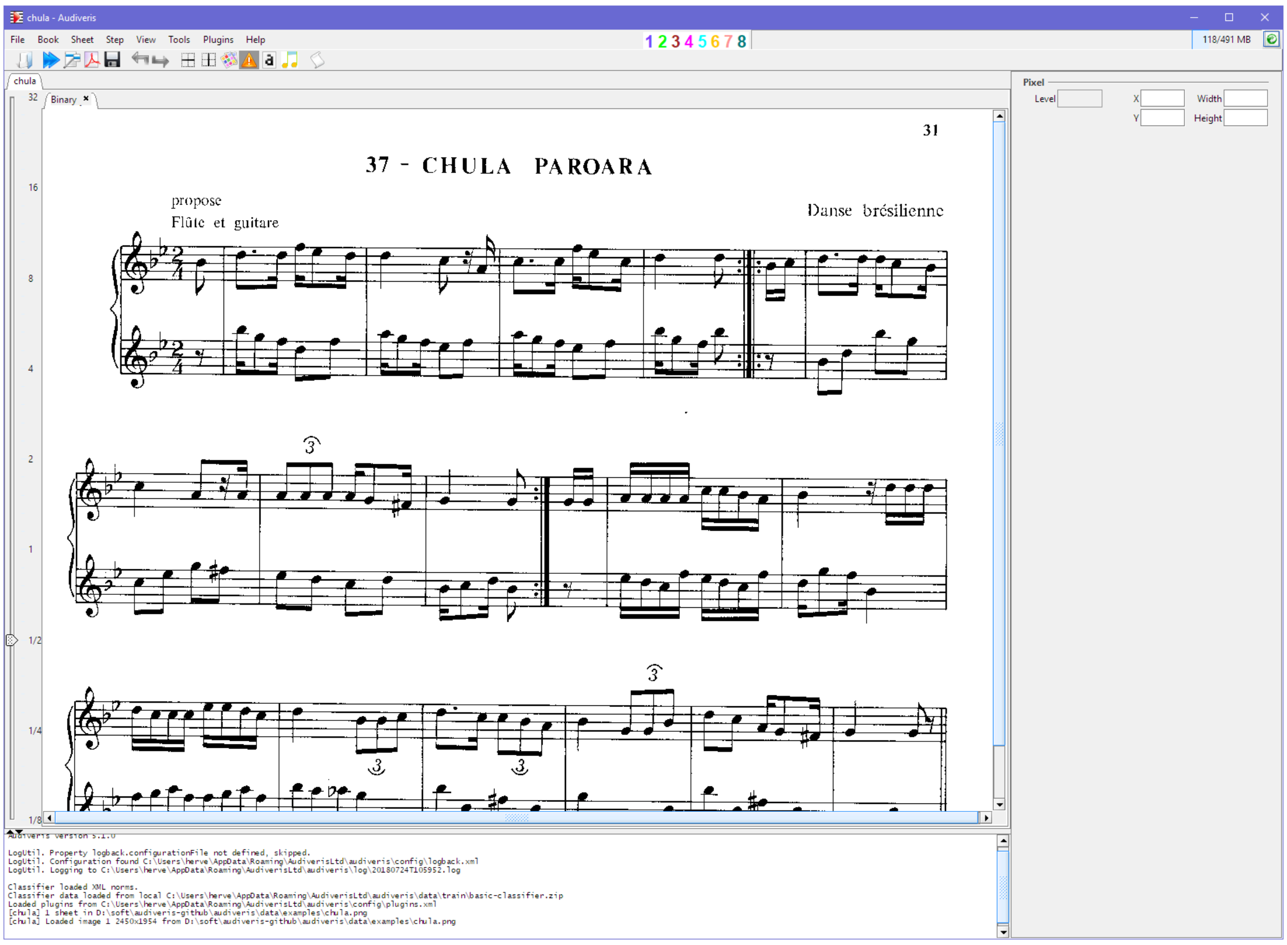
File Name: chula.png

Files of Type: Major image files (.bmp .gif .jpg .png .tiff .tif .pdf)

Another way to load an input file is to drag this file from the File Explorer and to drop it onto the Audiveris application window.

Audiveris accepts a variety of image formats as input, notably PDF, TIFF, JPG, PNG, BMP.

Here, we have selected the file `chula.png`, and the application window now displays the contained image. Actually, it’s a *binarized* version of the image:

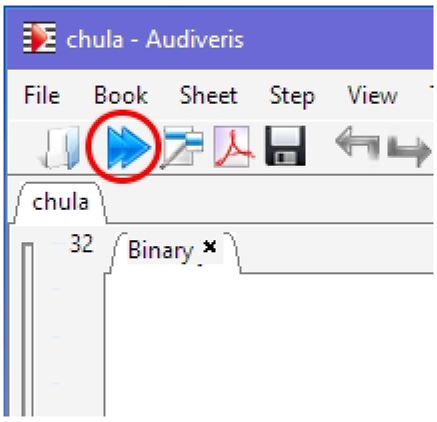


```
Audiveris version 5.1.0

LogUtil. Property logback.configurationFile not defined, skipped.
LogUtil. Configuration found C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\config\logback.xml
LogUtil. Logging to C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\log\20180724T105952.log

Classifier loaded XML norms.
Classifier data loaded from local C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\data\train\basic-classifier.zip
Loaded plugins from C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\config\plugins.xml
[chula] 1 sheet in D:\soft\audiveris-github\audiveris\data\examples\chula.png
[chula] Loaded image 1 2450x1954 from D:\soft\audiveris-github\audiveris\data\examples\chula.png
```

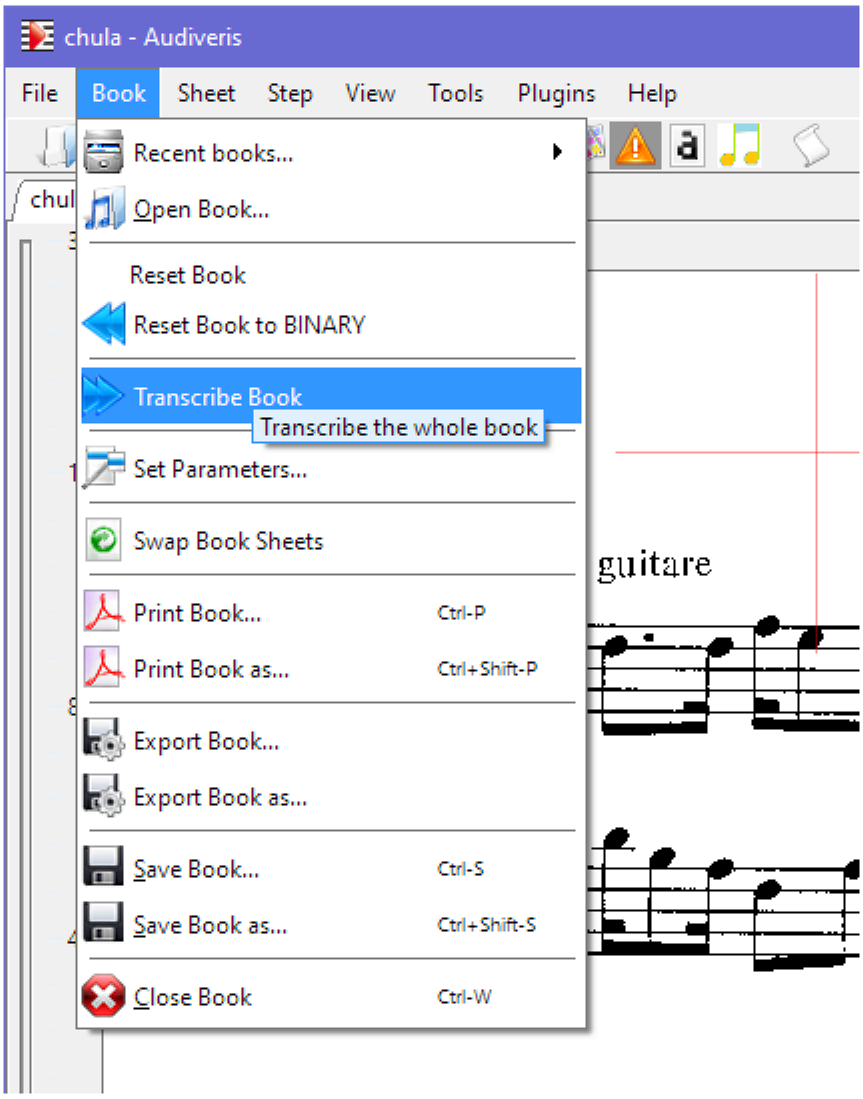
Transcribe

Transcription is the heart of OMR, the difficult process to infer high-level music information from low-level graphical data.

This process can be launched directly from the toolbar icon:



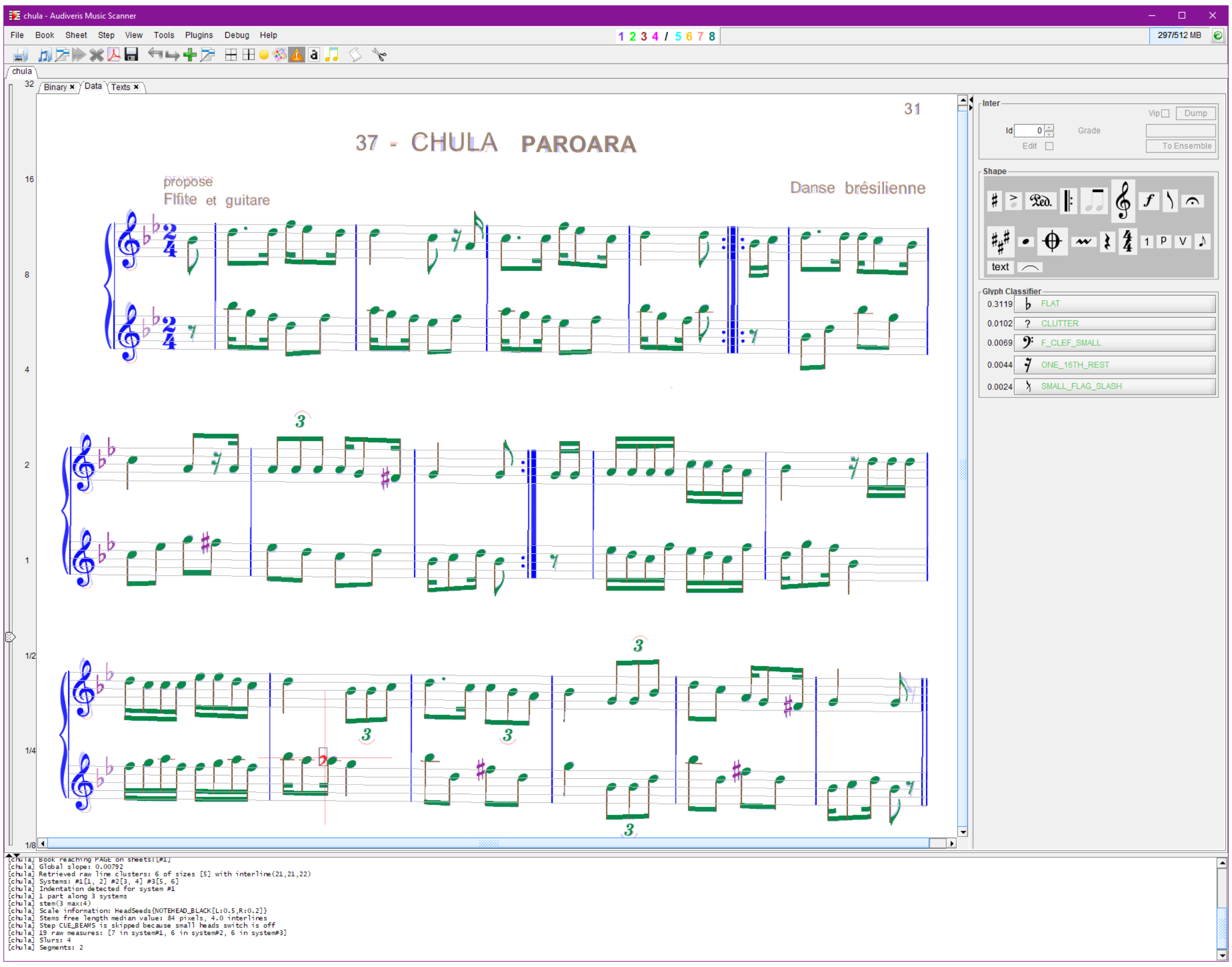
It can also be launched via the pull-down menu `Book → Transcribe book`:



This command applies to all images (sheets) in the input file (book).

There is also the pull-down menu `Sheet → Transcribe sheet` which applies only to the current sheet. In the simple case at hand, since we have just one image in the input file, the book-level and sheet-level commands are equivalent.

After some time, we get the following image of transcribed music:



0.3119	FLAT
0.0102	CLUTTER
0.0069	F_CLEF_SMALL
0.0044	ONE_16TH_REST
0.0024	SMALL_FLAG_SLASH

```
[chula] Book reaching PAGE on sheets:[#1]
[chula] Global slope: 0.00792
[chula] Retrieved raw line clusters: 6 of sizes [5] with interline(21,21,22)
[chula] Systems: #1[1, 2] #2[3, 4] #3[5, 6]
[chula] Indentation detected for system #1
[chula] 1 part along 3 systems
[chula] stem(3 max:4)
[chula] Scale information: HeadSeeds{NOTEHEAD_BLACK[L:0.5,R:0.2]}
[chula] Stems free length median value: 84 pixels, 4.0 interlines
[chula] Step CUE_BEAMS is skipped because small heads switch is off
[chula] 19 raw measures: [7 in system#1, 6 in system#2, 6 in system#3]
[chula] Slurs: 4
[chula] Segments: 2
```

Looking carefully at this result, we might detect a couple of mistakes:

- On the upper left, the word “Flûte” was OCR’d as “Flfite” [1]
- On the lower left, a flat sign was not recognized [2]
- On the lower right, a quarter rest was not recognized[3]

This could easily be corrected by manual actions, but let us simply ignore them for now to keep this example short.

Footnotes

[1] This is a French word, while the OCR language is English by default. See `Tools → Languages` ↩

[2] The underlying glyph collides with a ledger, resulting in poor grade. ↩

[3] This is due to an overlap with a flag item. ↩

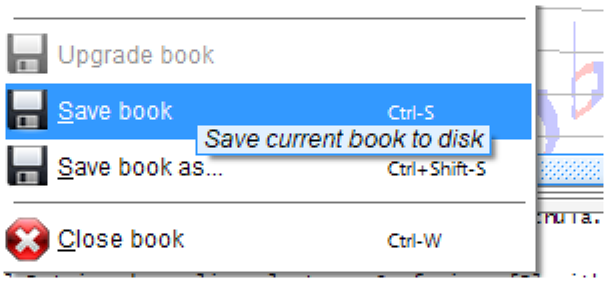
Export

Saving to `.omr` file

From the input image, the Audiveris OMR engine has gradually built specific information that we collectively refer to as “*OMR data*”.

The reference chapter on `.omr` files describes thoroughly how this OMR data is organized. For now, it is enough to know that we can save (and reload) this OMR information to/from disk (into a `.omr` file).

Saving can be done via the pull-down menu `Book → Save book` or via the `Ctrl+S` shortcut (`Command+S` for macOS):



In our “chula” example, this will result in a `chula.omr` file.

Saving the OMR job is not strictly necessary for our very small example.

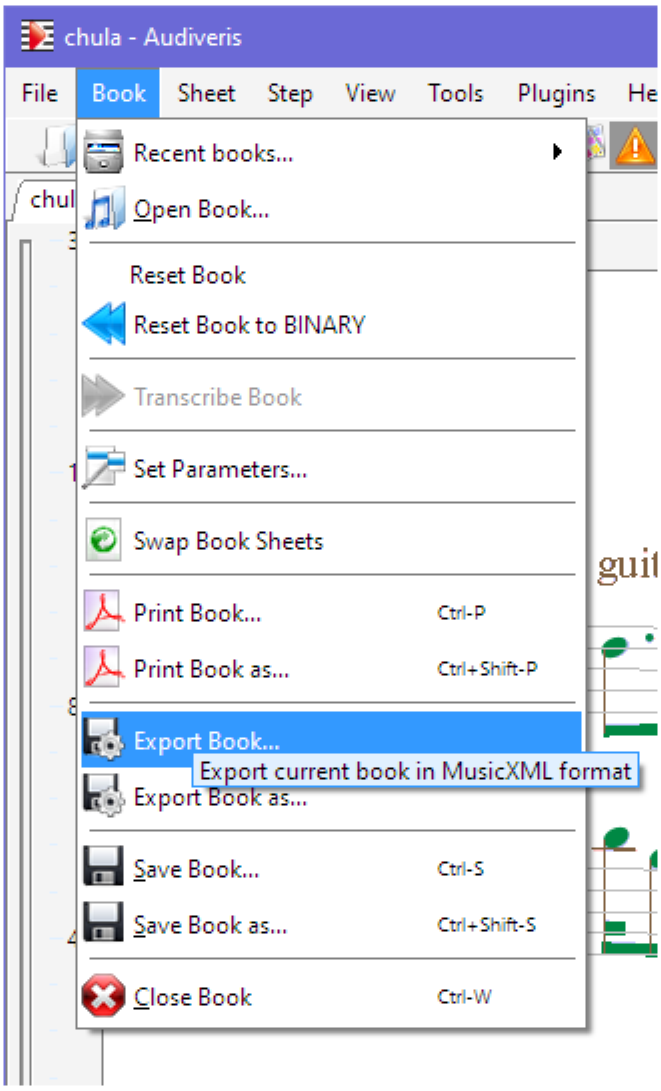
It is important to know that Audiveris can always restart from a saved `.omr` file, even from very old files, which allows to resume processing and user editing where they were stopped.

For sizable OMR jobs, this capability is essential.

Exporting to `.mxl` file

Right now, we are focused only on how to feed a music sequencer with music data it can easily import. And as of this writing, this is achieved by going through the *de facto* standard of **MusicXML**-formatted data.

This can be done via the pull-down menu `Book → Export book`:



From our “chula” example, this command produces a file named `chula.mxl` (the `.mxl` extension indicates a compressed MusicXML format).

Output location

Starting with the 5.4 release, the default policy is to put any output file next to the input file.

This default policy can be changed via the Preferences dialog.

We can also use the “`Save book as`” or the “`Export book as`” menu items to choose a specific output location and name.

`.omr` files vs. `.mxl` files

These files are not equivalent.

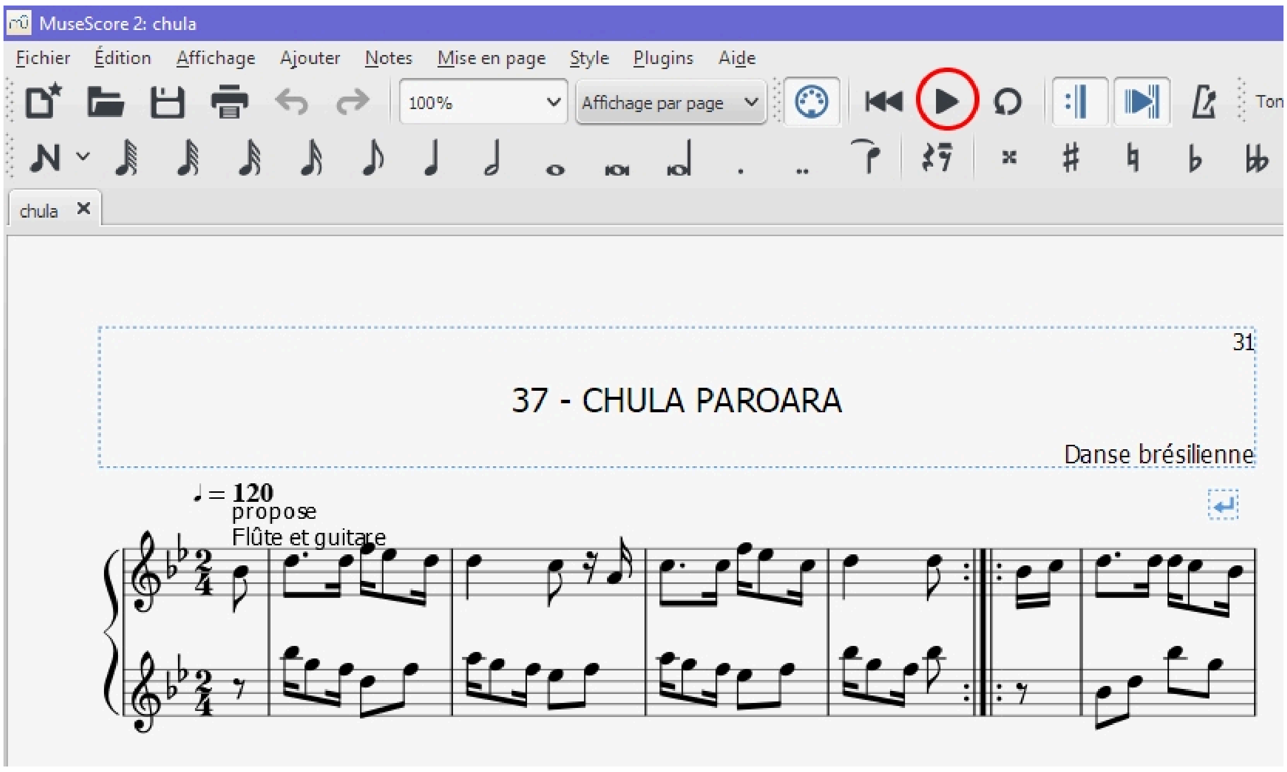
The export from OMR to MusicXML is *lossy*, since a large amount of OMR information can’t go into MusicXML.
A `.omr` file can always be used to regenerate the `.mxl` export, but the reverse is

not true.

> **NOTE**
>
> A good advice is to keep these `.omr` files – unless we are running out of disk space! :-) – because they represent a valuable source of OMR information, suitable for training newer versions of Audiveris (more on this later).

Play

Strictly speaking, this feature is not part of Audiveris, but rather pertains to an external program (a simple music sequencer or some high-level editor).
Many external programs can work on Audiveris MusicXML export, thanks to the *de facto* standard MusicXML exchange format.



Here, we have simply imported the `chula.mxl` MusicXML file into MuseScore.

Note that this music editor displays the imported music correctly. We just have to press the play button to listen to the music.

This is the end of our quick tour.

For a more thorough understanding of Audiveris, we can visit the next chapter: Main concepts.

Main concepts

The purpose of this chapter is to introduce some key notions that are used throughout the whole application and that the end-user needs to understand Audiveris correctly.

We will thus speak of:

- Physical and logical containment: Book of Sheets vs. Score of Pages,
- Pixels assemblies: Runs, Sections and Lags,
- Interpretations: Glyphs vs. Inters,
- Relations between interpretations: the Symbol Interpretation Graph (SIG).

TABLE OF CONTENTS

Book vs. Score

Book of Sheets

An image file fed into the OMR software contains one or several images. Typically `PDF` and `TIFF` formats support the notion of multi-image files while, for example, `JPEG` or `PNG` formats can deal only with single-image files.

For Audiveris, using the metaphor of a physical book made of several sheets of paper, this physical containment is modeled as one **Book** instance (corresponding to the input file) and a sequence of one or several **Sheet** instances (one sheet corresponding to one image).

Note that a sheet image may contain no music. This happens for example for a title or illustration or simply a blank sheet. In that case, the sheet will later be recognized as “*invalid*” (from the OMR point of view) and flagged as such.

Since Audiveris 5.3, we can split a book into smaller ones or, conversely, merge small books into a larger one. This feature is documented in the Split and merge section.

Score of Pages

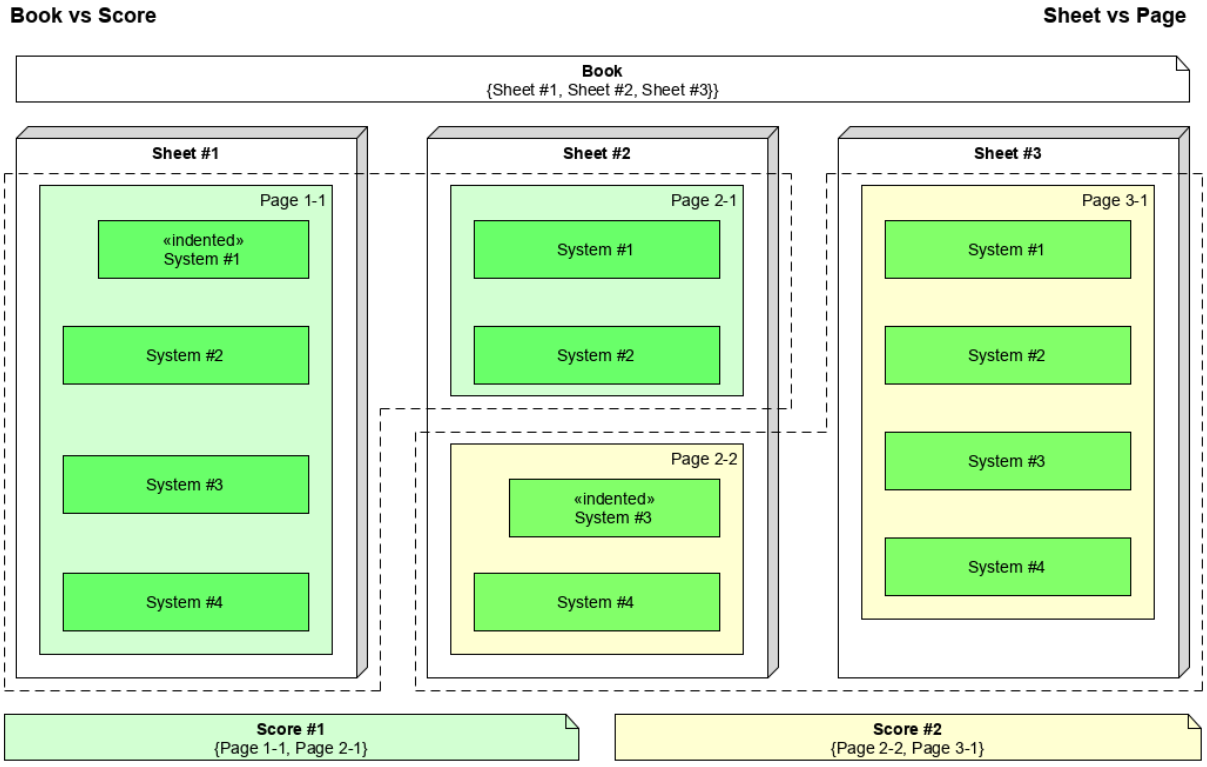
Looking at the musical content of a given sheet image, we can observe that staves are often gathered into systems (in which staves are played in sync), and that a given sheet image generally contains several systems (played one after the other).

A system may be left-indented with respect to the other systems in the image, to indicate the beginning of a movement. A non-indented system is assumed to belong to the same movement as the previous system (located just above in the current sheet or at the end of the previous sheet).

In Audiveris, this logical containment is modeled as one instance of **Score** per movement (since “Score” is the word used by MusicXML), the score containing a sequence of one or several **Page** instances.

Example

To make these concepts more concrete, let’s look at the following diagram of a hypothetical book:



In this diagram, we can see a book containing 3 sheets (certainly because the input `PDF` or `TIFF` file contained 3 images):

1. `Sheet` #1 begins with an indented system, which indicates the start of a movement (`Score` in MusicXML parlance). There is no other indented system, so this `Sheet` contains a single `Page`.
2. `Sheet` #2 exhibits an indentation for its system #3. This indented system ends the previous score and starts a new one. We have thus 2 `Page` instances in this `Sheet`.
3. `Sheet` #3 has no indented system and is thus composed of a single `Page`.

To summarize, in this book we have 2 scores that span over 3 sheets:

1. `Score` #1 is composed of:
   - Single `Page` #1 from `Sheet` #1, followed by
   - `Page` #1 from `Sheet` #2
2. `Score` #2 is composed of:
   - `Page` #2 from `Sheet` #2, followed by
   - Single `Page` #1 from `Sheet` #3

Rules

In the vast majority of cases, there is exactly one page per sheet.

The exceptions are as follows:

1. When an *indented system* appears anywhere in a sheet, it indicates the beginning of a new movement/score.
   This sheet, as in the picture above, then contains two pages or more, if the indented system is not the first one in the sheet.
2. When a sheet is *invalid* (i.e. containing no music), it contains no page.
   Moreover, an invalid sheet is considered as a *score break*:
   - It ends the last score in the previous valid sheet, if any.
   - The next valid sheet, if any, will start another score, even if it does not start with an indented system.

Pixels assemblies

The `BINARY` step transforms the input image into a black and white image. From this step on, the image will contain only black (foreground) pixels on a white background.

A (black) pixel is just a black square, of dimension 1 x 1, located at some point (x,y).

Depending on what the engine has to process (staff lines, stems, beams, etc), the same pixels can be viewed through one structure or another.

`Run` and `RunTable`

A horizontal (or vertical) contiguous sequence of pixels of the same color is called a horizontal (or vertical) `Run`.
In the same alignment, such `run` is followed by a `run` of the opposite color, and so on, until the image border is reached.

A `RunTable` is a rectangular area, made of sequences of `run`'s, all of the same orientation.
Typically, the whole binarized image can be considered, at the same time, as:

- a table of horizontal `run`'s
- a table of vertical `run`'s

`Section` and `LAG`

It can be interesting to transitively join adjacent (black) `run`'s of the same orientation, according to some compatibility rules.

Each such resulting assembly is called a `Section`.

Typical compatibility rules are:

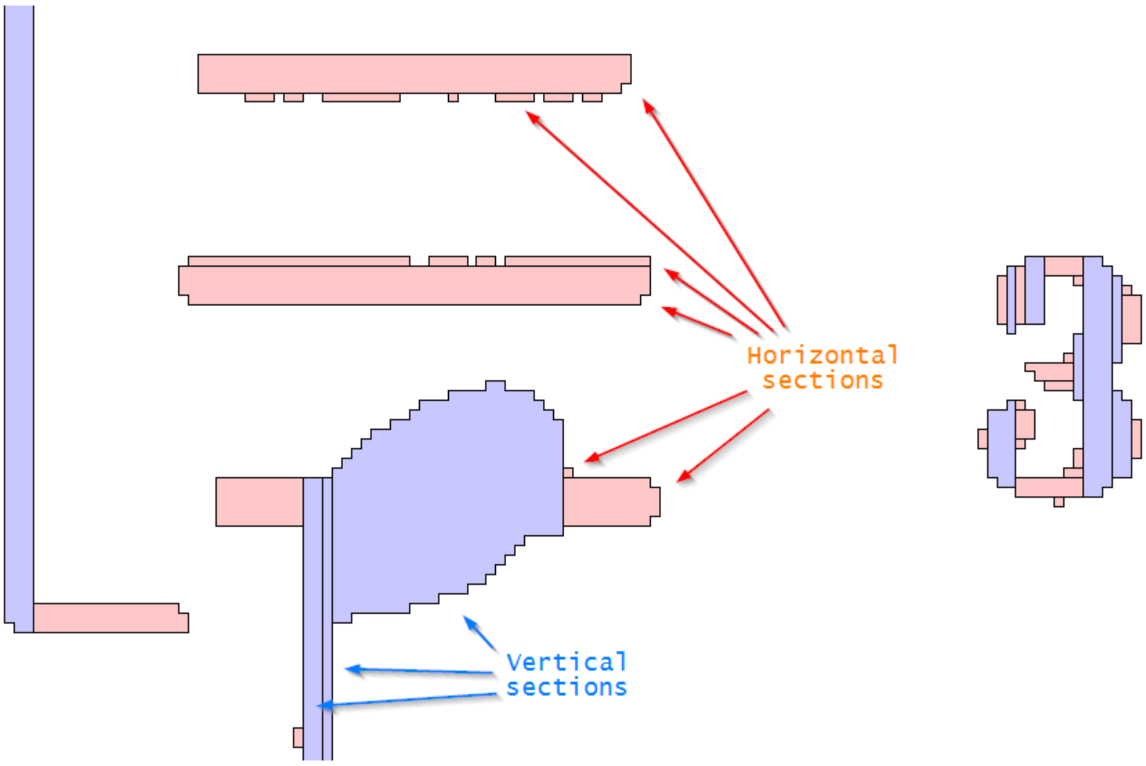
- Maximum difference in `run` lengths
- Maximum ratio of difference in `run` lengths
- Maximum shift on each `run` end

- Void rule (no check, except adjacency)

Sections are gathered into `LAG`’s (**L**inear **A**djacency **G**raphs).

Just like a `RunTable` gathers `Run`’s of the same orientation, a `LAG` gathers `Section`’s of the same orientation.

Sections example



The picture above can be displayed once the `GRID` step has been performed. We select the “section” view via the `View → Switch selections` pull-down menu or the `F11` function key.

Based on the maximum staff line thickness (previously determined by the `SCALE` step), this picture combines sections from two different `LAG`’s:

1. From the vertical `LAG`, all the (vertical) sections with length greater than the maximum line thickness are displayed in pale blue.
2. From the horizontal `LAG`, the remaining pixels are organized in (horizontal) sections and displayed in pale pink.

Filament

A `Filament` is a dynamic assembly of sections, long and thin, likely to represent lines.

The engine uses:

- horizontal filaments to detect staff lines and ledgers alignments,
- vertical filaments to detect stems and legs of endings.

`Glyph` vs. `Inter`

`Glyph`

A `Glyph` is nothing more than a set of foreground (black) pixels in a sheet binary image.

It carries no shape.

It is not related to a staff. It does not even belong to a system. The reason is there is no reliable way to assign a glyph located in the “gutter” between two systems or two staves: does it belong to the upper or the lower system/staff?

These restrictions on `Glyph` don’t apply to glyph interpretations (a.k.a. `Inter`).

`Inter`

An interpretation, or `Inter` for short, is meant to formalize one reasonable interpretation of a `Glyph`.

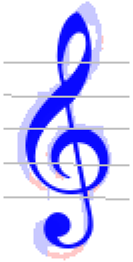
There may be several reasonable interpretations for a given glyph and, in many cases, the OMR engine cannot immediately decide on the right interpretation among these mutually exclusive interpretations. This decision will then be postponed until later down in the OMR process, when additional information (such as the discovery of other `Inter` instances located nearby) becomes available and helps clarify the configuration.

As opposed to a `Glyph`, an `Inter` belongs to a system and is often related to a staff.

It carries a *shape* and a *grade* in [0..1] range, which can be considered as the probability for the interpretation to be a true positive.
This grade is an interpretation *intrinsic* grade, only based on the glyph at hand in isolation (this grade is often provided by the glyph classifier).

Later, the `Inter` will generally be assigned a *contextual* grade, based on the `Inter` intrinsic grade and the supporting relations with other `Inter` instances nearby.

Typical display example

View	`Inter` **over** `Glyph`
	Here we have a Treble Clef `Inter` displayed in dark blue. Its related `Glyph`, using pale blue and pink colors, is mostly hidden behind.

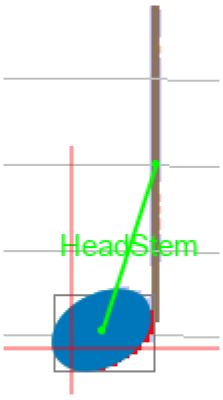
Symbol Interpretation Graph (SIG)

Relation

A *Relation* instance formalizes a relationship between a source `Inter` instance and a (separate) target `Inter` instance.

There are 3 kinds of relation:

- A *positive* relation represents a **mutual support** between two `Inter` instances.

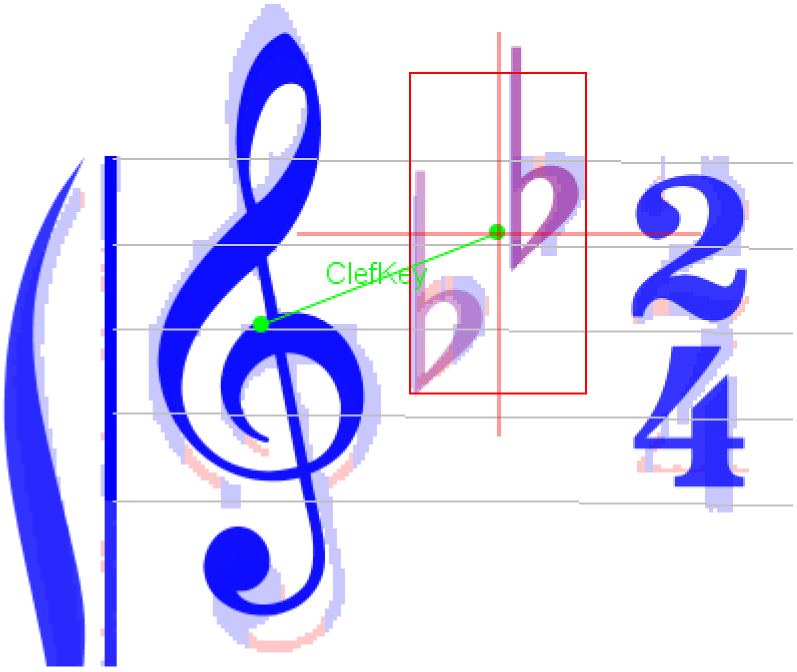
  Here we have a typical example: a filled head interpretation and a stem interpretation nearby with a suitable *HeadStemRelation* between them, shown as a “*HeadStem*“-labelled green segment (the “*Relation*” suffix is always omitted in relation name display).
  Support relations *increase* the contextual grade of their linked `Inter` instances. In this way, even rather low-quality interpretations, when well combined through supporting relations, may end up with acceptable contextual grade.

- A *negative* relation represents a **mutual exclusion** between two `Inter` instances. Typically, two different interpretations for the same underlying glyph will compete with one another and will thus be linked by an *Exclusion* relation.
  An exclusion tells the engine that the two `Inter` instances cannot coexist in the final configuration, so at least one of them will be discarded at some point in the transcription process.

- A *neutral* relation is neither positive nor negative, it just conveys information between two `Inter` instances.
  For example a head-chord is an ensemble composed of one or several heads and often one stem. There is one *Containment* relation between the chord ensemble and each of its heads members. If the chord has a stem, there is a *ChordStemRelation* between chord and stem (along with a supporting *HeadStemRelation* between each head and the stem).

The image below shows a higher level of relation:

- We can see two `Inter` instances (a treble clef followed by a 2-flat key) linked by a supporting *ClefKeyRelation* instance.
- This *ClefKeyRelation* exists because the vertical positions (pitches) of the flat signs configuration in this key (B then E) are compatible with a treble clef.
- If ever there was a competing bass clef candidate, there would be of course an *Exclusion* between the two competing clef candidates, plus another *Exclusion* between the bass clef and this key (because the vertical positions of this key are not compatible with a bass clef).



SIG

A **S**ymbol **I**nterpretation **G**raph (SIG), is simply a graph with `Inter` instances as vertices and Relation instances as edges.

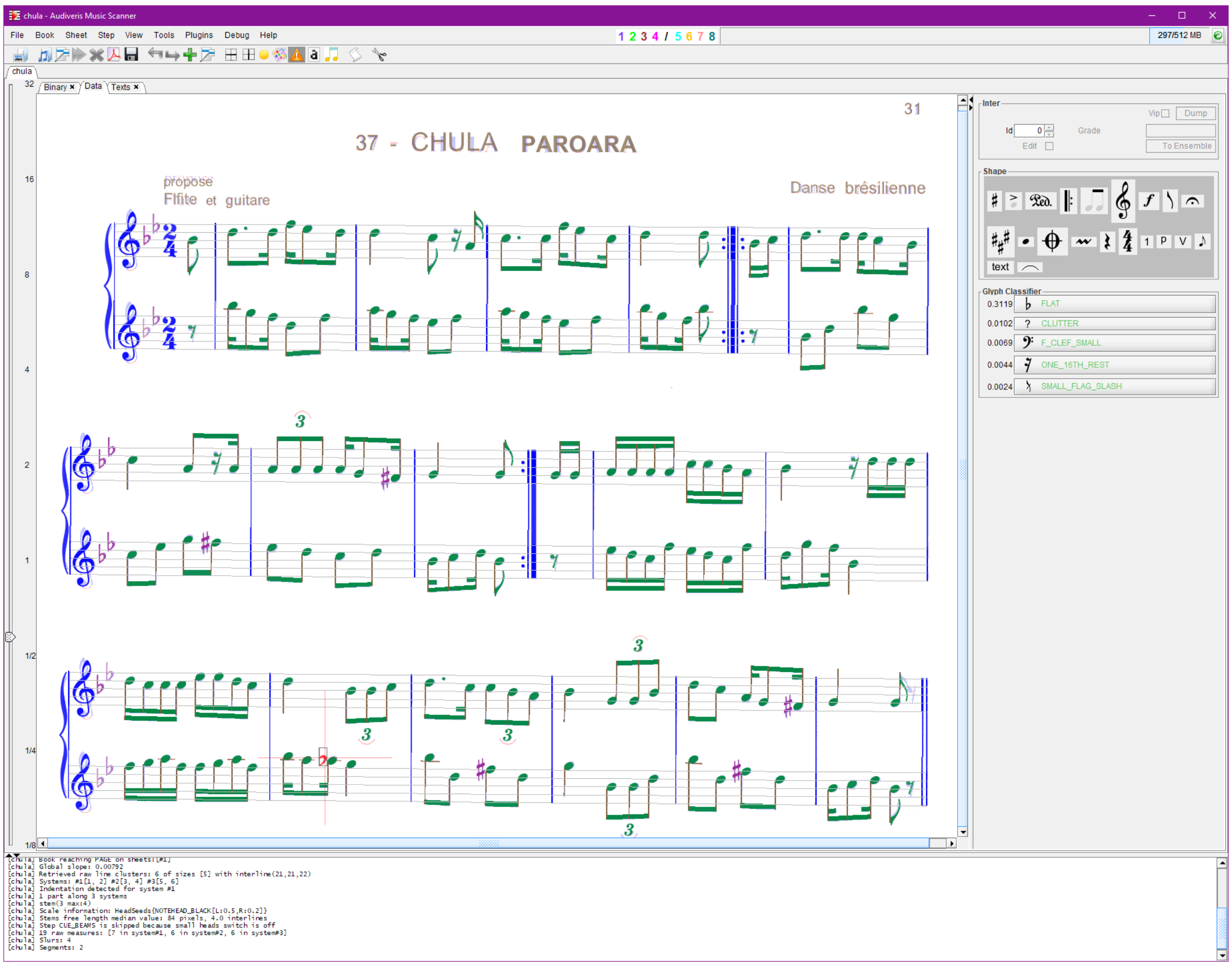
The SIG plays a central role in Audiveris V5. Its main purpose is to host all candidate Interpretations and to formalize and manage the Relations (mutual exclusions, supporting relations) within the population of candidate interpretations.

There is one SIG per system, and at precise points in the OMR engine pipeline (the REDUCTION step and the LINKS step), each SIG is *reduced* so that all exclusions are resolved and no `Inter` with weak contextual grade remains in its graph.

Main window

This is the window that appears when Audiveris is run in the interactive mode (which is the standard mode, unless the `-batch` option is specified on the command line).

From this central window, we can drive the transcription process and edit the results.



```
[chula] Book reaching PAGE on sheets:[#1]
[chula] Global slope: 0.00792
[chula] Retrieved raw line clusters: 6 of sizes [5] with interline(21,21,22)
[chula] Systems: #1[1, 2] #2[3, 4] #3[5, 6]
[chula] Indentation detected for system #1
[chula] 1 part along 3 systems
[chula] stem(3 max:4)
[chula] Scale information: HeadSeeds{NOTEHEAD_BLACK[L:0.5,R:0.2]}
[chula] Stems free length median value: 84 pixels, 4.0 interlines
[chula] Step CUE_BEAMS is skipped because small heads switch is off
[chula] 19 raw measures: [7 in system#1, 6 in system#2, 6 in system#3]
[chula] Slurs: 4
[chula] Segments: 2
```

This main window is composed of 3 panels: sheet, boards and events.

Sheet

This is the large panel on the left side.

- The **Gray** tab, when available, presents the original image using gray values.

- The **Binary** tab presents the input image binarized into black and white colors.
- The **Data** tab presents the objects (sections, glyphs and inters) extracted from the image. In this `Data` tab, the staff lines are logically removed and drawn as thin lines.

All tabs, except the `Data` tab, can be manually closed. Most can be re-opened via the `Sheet` pull-down menu.

Boards

The right panel is a vertical set of boards. They provide information and editing functions.

Only basic boards are displayed by default, the others are hidden. A right click in this column allows to hide or display any board available for the current sheet tab.

Events

The lower panel is a log of the main events that occurred so far.

More details are available in the Audiveris log file (the precise path to the log file is displayed at the top of this events panel).

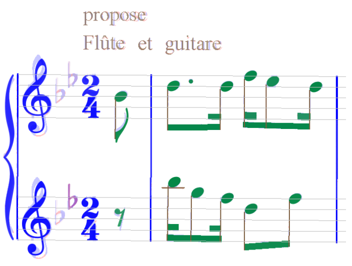
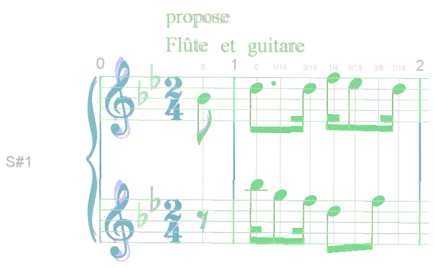
TABLE OF CONTENTS

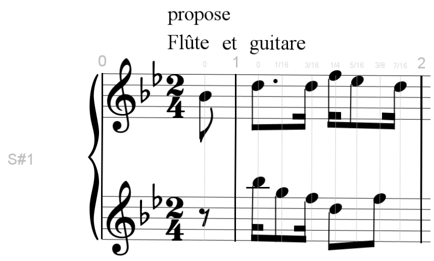
Data display modes

In the sheet panel we can choose between 3 display modes, that are effective in the **Data** tab:

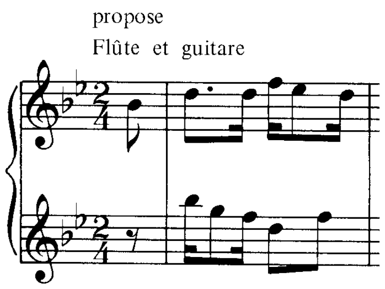
- ♫ The ***physical*** mode displays in the background the sheet sections of pixels (pale blue for vertical sections, pale pink for horizontal sections) and in the foreground the current detected inters colorized according to their recognized shape and quality grade.
- ♫ The ***combined*** mode is a combination of the physical and logical layers. It displays the logical interpretations in a translucent manner on top of the physical pixels, to ease the visual detection of any discrepancies.
- ♫ The ***logical*** mode displays only the logical score entities (inters). It represents the current transcription of the original image, annotated by informations such as system number, measure number, time slot offset, etc.

Using the menu `View → Switch layers` or the **F12** function key or the dedicated toolbar icon (♫/♫/♫), we can cycle through these 3 different modes: Physical / Combined / Logical.

Mode	Data tab
♫ Physical mode	
♫ Combined mode	

Mode	Data tab
♫ Logical mode	

The other tabs are not impacted by the display mode.
Notably, the **Binary** tab (which was mode-sensitive in previous Audiveris versions) now remains unmodified, so that it can instantly be used as a reference via a simple click on its tab:

No mode impact	Binary tab
	

Pan and zoom

Moving

The sheet image is usually larger than the window where the sheet is displayed. We can move the display over the sheet using different means:

- By moving the scroll bars,
- By keeping the mouse left button pressed, and moving the selection point close to a border of the display,
- By keeping both mouse buttons pressed, and dragging the image with the selection point,
- By pressing down the mouse wheel and dragging the image with the selection point,
- By rolling the mouse wheel to move the display up or down,
- By pressing on one of the 4 arrow keys to move the display in related direction,
- By pressing on one of the 4 arrow keys, while `Shift` is held down, to move pixel by pixel.

Zoom

It can be adjusted in the range [1:8 to 32:1] by several means:

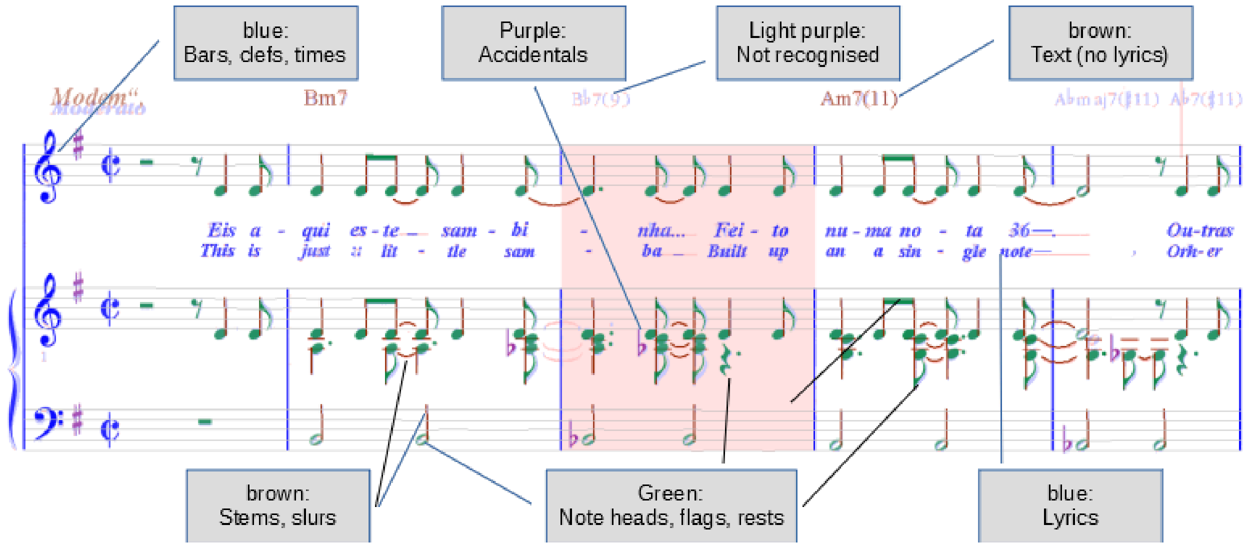
- By using the vertical slider located on the left side of the sheet window.
- By using the mouse wheel while holding the `Ctrl` key pressed.
- On the numeric keypad:
  - Key `+` to zoom in,
  - Key `-` to zoom out,
  - Key `0` to reset zoom.
- On the keyboard:
  - Key `Ctrl+ =` (as well as `Ctrl+Shift =`) to zoom in,
  - Key `Ctrl+ -` (as well as `Ctrl+Shift -`) to zoom out,
  - Key `Ctrl+ 0` (as well as `Ctrl+Shift 0`) to reset zoom.

- By using a rectangular “lasso” while holding both keys `Ctrl+Shift` pressed. When releasing the mouse, the zoom will be adjusted so that both rectangle sides become fully visible.
- By using the predefined buttons and , we can fit to the sheet’s width or height, respectively.

When zooming in or out, the display remains focused on its current selection, if any.

Entity colors

The sheet window uses different colors to display the entities and the kind of entity that was recognized:



In this example the *background* of the third measure stack is colored in pink because the rhythm check has failed for this measure (the first note of the upper voice was mistaken for a quarter instead of a half note).

The following *foreground* colors are used for interpretation items in this window:

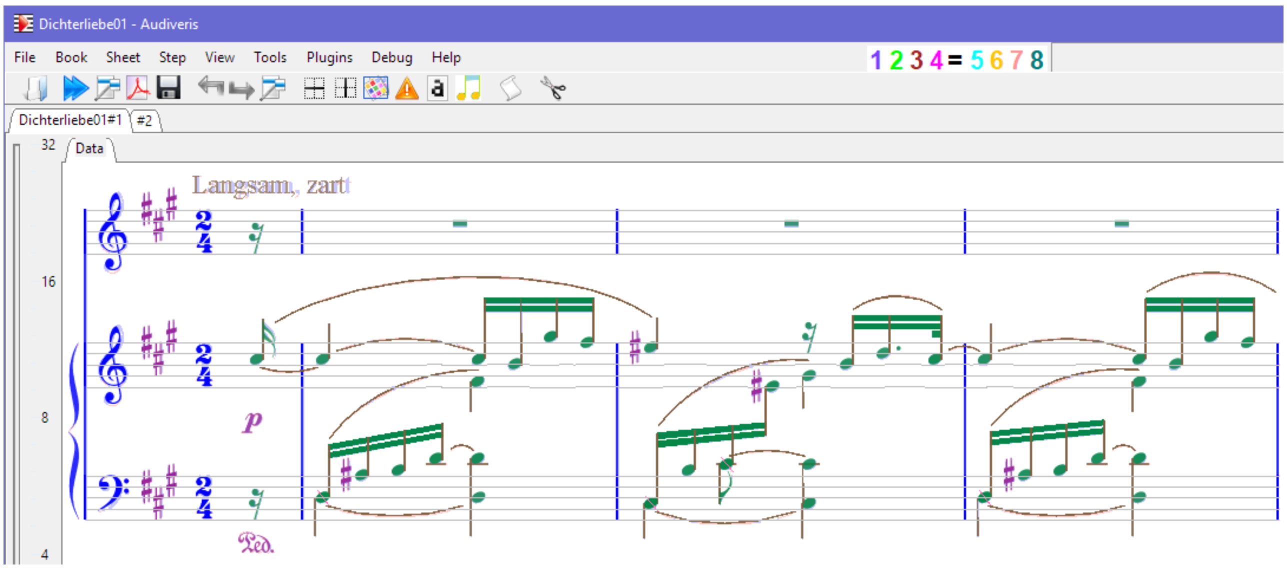
- blue: bars, clefs, times, lyrics
- purple: accidentals, articulations
- pale blue / pale pink: non recognized elements (vertical sections / horizontal sections)
- brown: stems, slurs, normal text (not lyrics)
- green: note heads (also small heads and augmentation points), flags, rests
- red: elements in some abnormal status; typically some needed connection is missing (e.g. black note head without stem)

The colors help the user for a first glance verification that the transcription was successful or where some corrections will be needed later on.

Tutorials / Main window / Voice colors

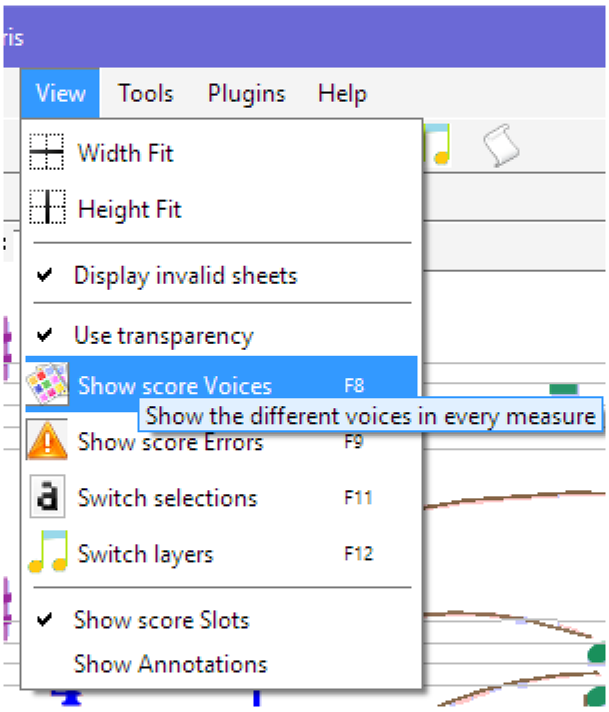
Voice colors

By default, as described in the previous section, each score entity is displayed using a color determined according to the entity kind:


Dichterliebe01 - Audiveris
File Book Sheet Step View Tools Plugins Debug Help
1 2 3 4 = 5 6 7 8
Dichterliebe01#1
#2
Data
32
16
8
4
Langsam, zart
p
Ped.

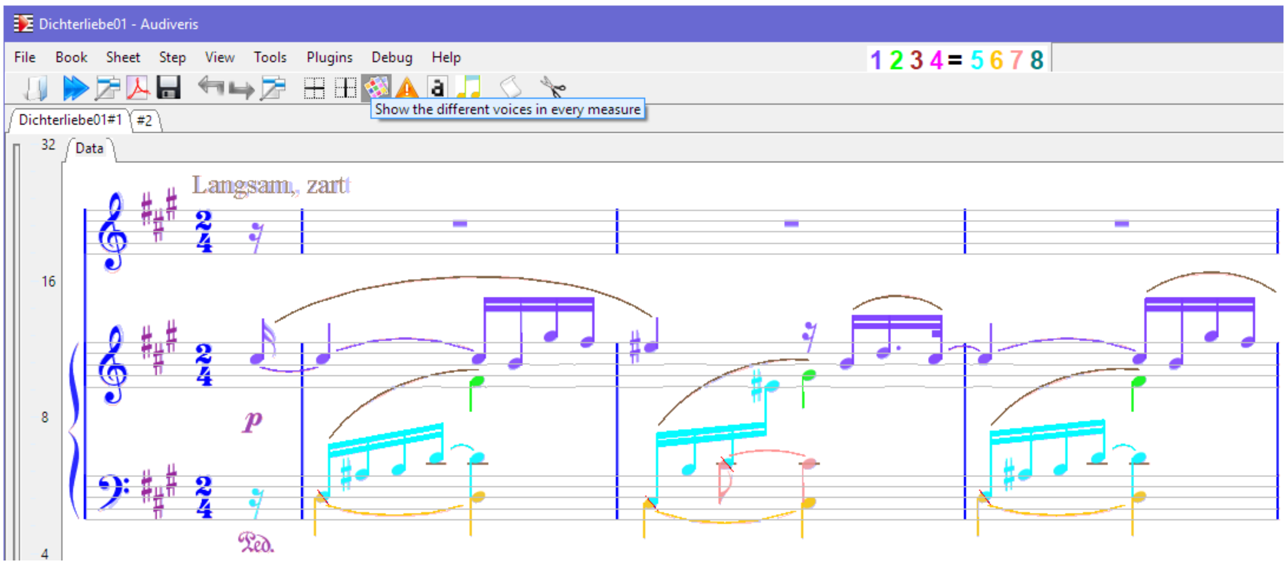
But we can decide to focus on **voices**, rather than entity **kinds**, and thus choose to display each entity according to its voice, if applicable. This feature is reported to ease the detection of wrong voice assignment.

To do this, we use the pull-down menu `View → Show score voices` or the related toolbar button:



Show the different voices in every measure

This results in the following display:



Within any given part, voice numbers (and thus colors) are assigned as follows: 1 2 3 4 / 5 6 7 8

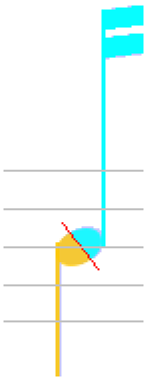
- The voices *starting* on the upper staff use numbers 1 through 4,
- The voices *starting* on the lower staff use numbers 5 through 8.

Shared heads

Note that some note heads can be *shared* between two chords. In the example above, this is the case in the last staff, for the starting head of each measure, except the first one.

In such canonical case, the chords involved are the chord below on the left and the chord above on the right.

To indicate the ***shared*** aspect of such head, a small diagonal red segment is drawn across the head, to indicate a logical split of the shared head.



Here, voices are colorized, thus each head *'half'* appears with its own voice color.

How-to guides

This `How-to` chapter guides the user around specific tasks, presented in a progressive manner.

TABLE OF CONTENTS

Main features

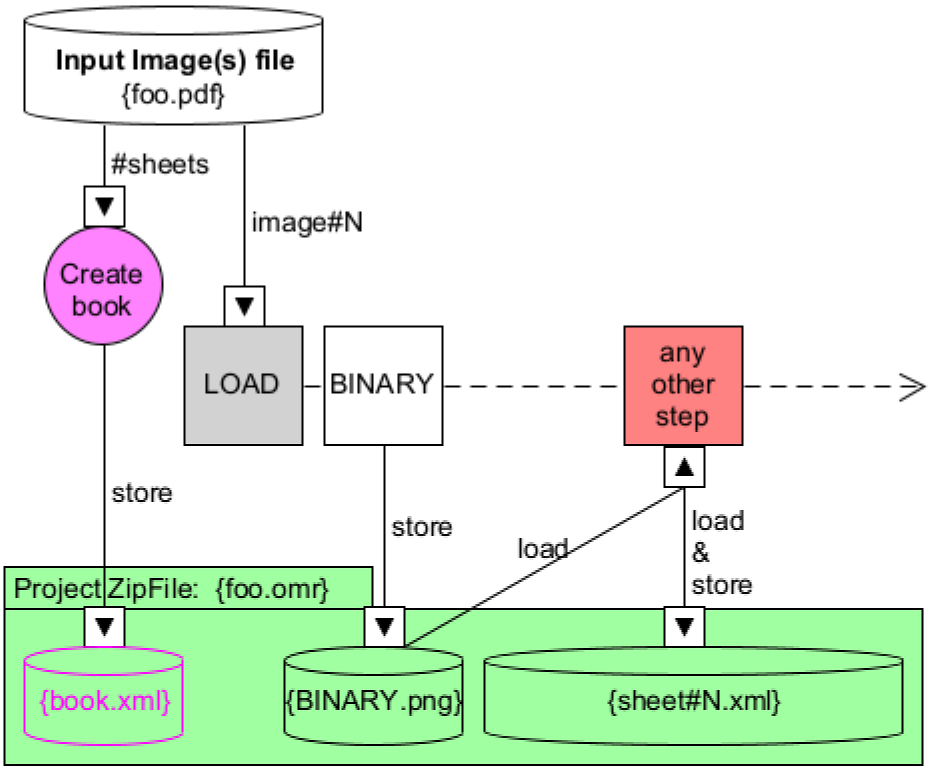
This chapter presents the main features of the Audiveris application, especially the pipeline of the OMR engine, from input to outputs.

TABLE OF CONTENTS

Pipeline

Global book workflow

When working on a book, the Audiveris V5 OMR engine can process any sheet of the book independently of the others. Only the final gathering of sheets results, which comparatively is a very fast action, is performed at book level.



Global Book Workflow

The diagram above presents the typical workflow for an example input file, named `foo.pdf`:

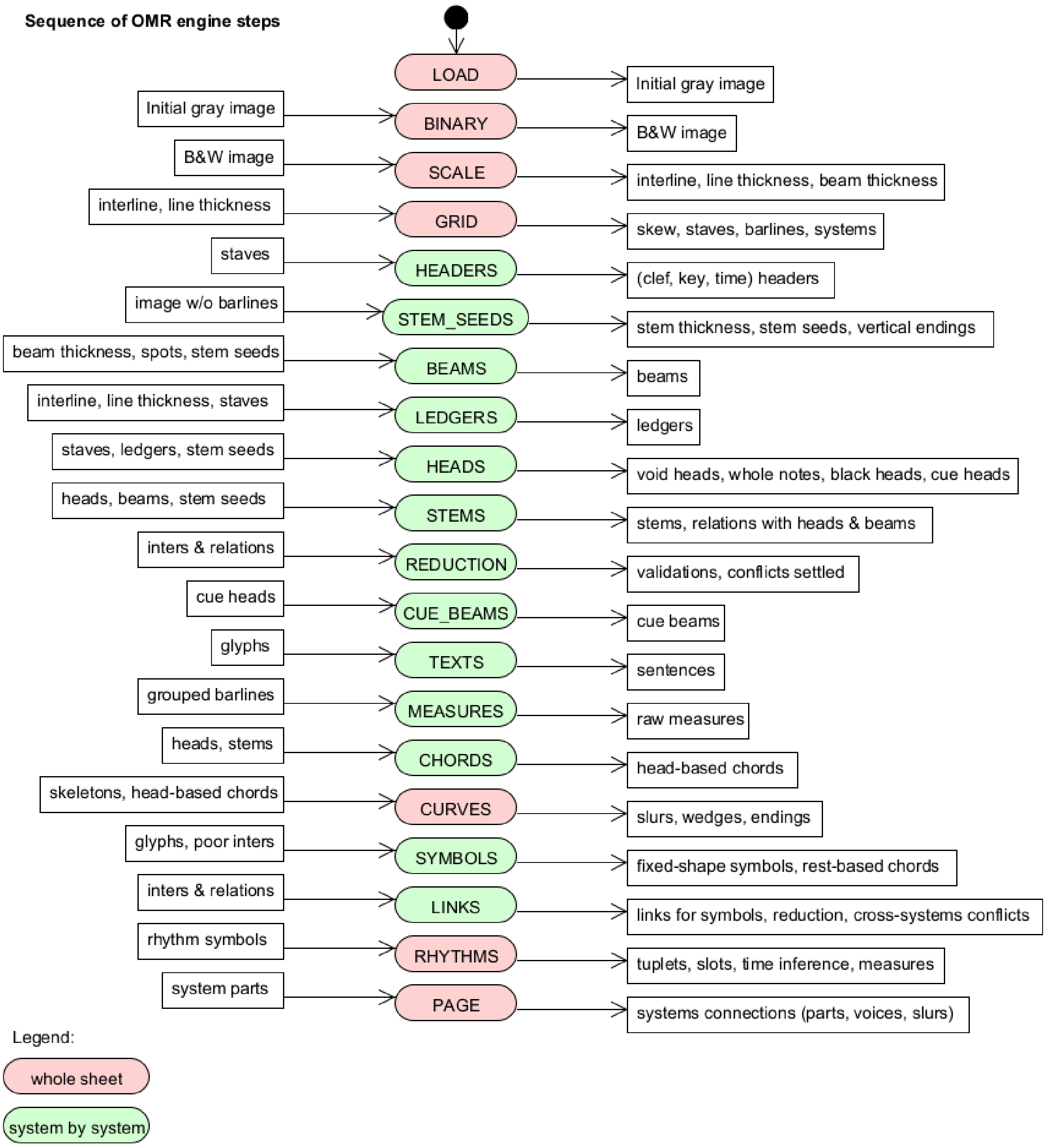
1. When opening the `foo.pdf` input file, Audiveris creates a Book instance.
2. It then detects how many images the input file contains, and allocates one sheet (just a sheet “stub” actually) for each contained image.
3. When processing a given sheet, the corresponding image is loaded from the input file, and the OMR pipeline is applied on the sheet.
4. At any time, when saving the project, all the book and sheets OMR information is saved into the `foo.omr` project file.

> **TIP**: Audiveris V5 can accommodate a book of hundreds of sheets. To save on memory, especially during long interactive sessions, we can ask Audiveris to transparently swap all book sheets to disk (except the current sheet). This is done via the pull-down menu `Book → Swap book sheets`.

Sheet pipeline

The processing of a given sheet by the OMR engine is done via a pipeline of some 20 steps applied, one after the other, on sheet OMR data.

Here below is the sheet pipeline sequence, with the main inputs and outputs of every step:


Sequence of OMR engine steps
LOAD
Initial gray image
Initial gray image
BINARY
B&W image
B&W image
SCALE
interline, line thickness, beam thickness
interline, line thickness
GRID
skew, staves, barlines, systems
staves
HEADERS
(clef, key, time) headers
image w/o barlines
STEM_SEEDS
stem thickness, stem seeds, vertical endings
beam thickness, spots, stem seeds
BEAMS
beams
interline, line thickness, staves
LEDGERS
ledgers
staves, ledgers, stem seeds
HEADS
void heads, whole notes, black heads, cue heads
heads, beams, stem seeds
STEMS
stems, relations with heads & beams
inters & relations
REDUCTION
validations, conflicts settled
cue heads
CUE_BEAMS
cue beams
glyphs
TEXTS
sentences
grouped barlines
MEASURES
raw measures
heads, stems
CHORDS
head-based chords
skeletons, head-based chords
CURVES
slurs, wedges, endings
glyphs, poor inters
SYMBOLS
fixed-shape symbols, rest-based chords
inters & relations
LINKS
links for symbols, reduction, cross-systems conflicts
rhythm symbols
RHYTHMS
tuplets, slots, time inference, measures
system parts
PAGE
systems connections (parts, voices, slurs)
Legend:
whole sheet
system by system

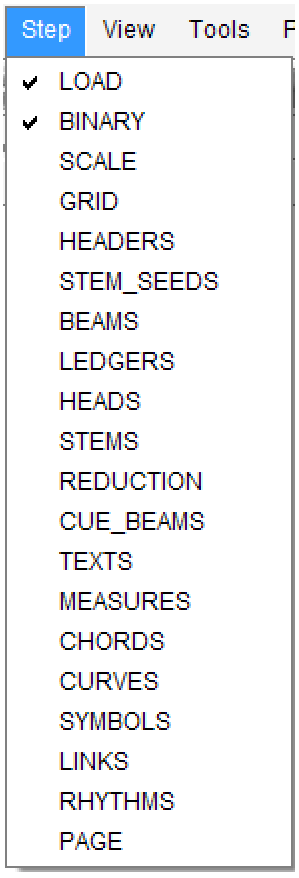
NOTE

Each of these 20 steps is detailed in the dedicated explanation pages.
There is no need to go through all of them in a first reading of this handbook.
We can just keep in mind that this information is available and is likely to help us understand how each of these steps works.

Driving the pipeline

A sheet step is like a mini-batch applied on the sheet data, and this is the smallest increment that the OMR engine can perform.

In the selected sheet, we can decide to move the pipeline forward until a target step. To do so, we select the target step in the pull-down `Step` menu:



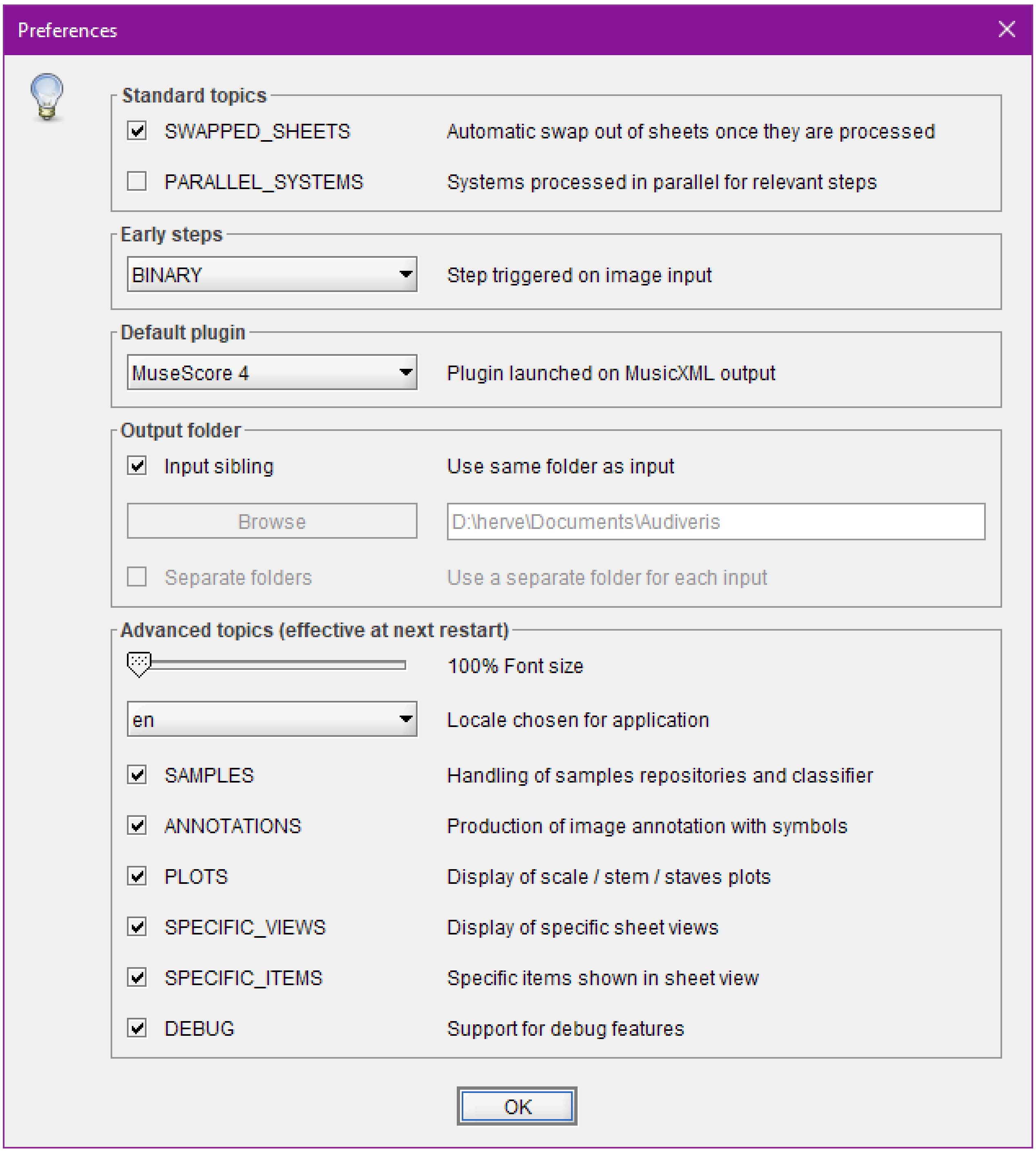
Note that selecting the pull-down menu `Sheet → Transcribe sheet` is just another way of selecting the pull-down menu `Step → PAGE`.

Be careful, we cannot move the pipeline directly backward.
But there are two indirect workarounds:

- We can select a target step that has already been performed:
    1. we are first prompted for confirmation,
    2. the sheet data is then reset to its initial status – the gray image if available, otherwise the binary image,
    3. and all necessary steps are re-performed up to the target step.
- We can also abandon the book and reload it from a previously saved status.

Preferences UPDATED IN 5.9

The `Tools → Preferences` pull-down menu opens this dialog, focused on on the handling of a few user preferences.
It operates at a much higher level than the direct handling of application constants.



Preferences

Standard topics

☑ SWAPPED_SHEETS — Automatic swap out of sheets once they are processed

□ PARALLEL_SYSTEMS — Systems processed in parallel for relevant steps

Early steps

BINARY — Step triggered on image input

Default plugin

MuseScore 4 — Plugin launched on MusicXML output

Output folder

☑ Input sibling — Use same folder as input

Browse — D:\herve\Documents\Audiveris

□ Separate folders — Use a separate folder for each input

Advanced topics (effective at next restart)

100% Font size

en — Locale chosen for application

☑ SAMPLES — Handling of samples repositories and classifier

☑ ANNOTATIONS — Production of image annotation with symbols

☑ PLOTS — Display of scale / stem / staves plots

☑ SPECIFIC_VIEWS — Display of specific sheet views

☑ SPECIFIC_ITEMS — Specific items shown in sheet view

☑ DEBUG — Support for debug features

Beside standard features, the dialog also presents a set of advanced features that impact the Audiveris user interface and thus require an application restart.

TABLE OF CONTENTS

Standard processing options NEW IN 5.9

These options impact the way the engine can work, especially for demanding scores.

- The **SWAPPED_SHEETS** option determines whether the engine should save a sheet to disk and then remove it from memory after processing.
  This is particularly useful when transcribing a book with many sheets, as it keeps memory consumption low without significantly impacting the overall transcription time.
  This option is enabled by default.

- The **PARALLEL_SYSTEMS** option determines whether the engine should process all the systems for a sheet in parallel.
  This only applies to the steps where parallelizing the systems is relevant, which is 13 of the 20 defined steps. The overall transcription time saving is approximately 25%.
  A drawback of this approach is that the results are not strictly reproducible, since the systems compete for allocating and naming entities like Inter instances.
  This option is disabled by default.

Early steps

This box allows to define which step is automatically trigerred on an input file.

Default plugin

This allows to interactively choose a new default plugin among the declared ones, since by default the first declared plugin is set as the default one (See the Plugins section).

Output folder

These boxes govern where output files are located by default.

- **Input sibling**: If set, all outputs are stored in the same folder as the input file (unless the `-output` option is specified on the command line).
- **Browse**: Allows to define a default output folder.
- **Separate folders**: If set, all outputs related to a given input file are stored in a *separate* folder, created according to the input file name without its extension.

For further explanations, see the section on Standard folders.

Global font ratio

The slider allows to select a larger font size used throughout the application views.

The new ratio is applied at the next restart.

Locale

We can pick up a different user language.
As of this writing, the available locales are:

- **en** (English), the default
- **fr** (French), not yet fully implemented…[1]

The new locale is applied at the next restart.

Advanced options

Each of these options can gather several related features.

- **SAMPLES** deals with sample repositories and classifier training.

- **ANNOTATIONS** deals with the production of symbol annotations.
- **PLOTS** deals with the display of plots for scale, stem, staff or header projections.
- **SPECIFIC_VIEWS** deals with specific sheet tabs (staff free, staff-line glyphs).
- **SPECIFIC_ITEMS** deals with the display of specific items on views (attachments, glyph axis, …)
- **DEBUG** deals with many debug features (notably browsing the book internal hierarchy).

An **application restart** is needed to take any modified selection into account.

Footnotes

[1] If you are willing to add another locale, please post a message on the Audiveris discussion forum. ↩

OCR languages NEW POLICY IN 5.4

Audiveris delegates text recognition to the Tesseract OCR library.

Tesseract OCR is potentially capable of processing text in over a hundred languages.
But, for a language to be effectively supported:

1. A specific data file (containing learning materials, a dictionary, etc. for that language) must be installed locally.
2. This language must be specified as default, or for the book, or just for the sheet in question.

For example, the processing of an input score using the English language (code: `eng`) requires that:

1. the `eng.traineddata` file is installed beforehand;
2. the `eng` code is specified for the input score.

TABLE OF CONTENTS

No preinstalled languages

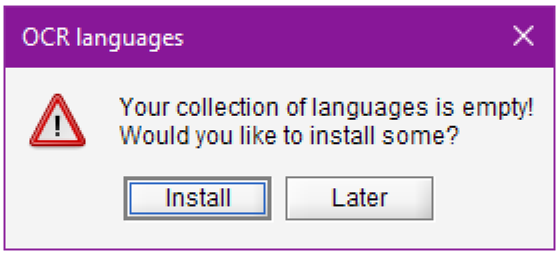
Recent history:

- In version 5.4, no languages were installed by default, except for the old Windows installer without JRE, which preinstalled English.

-

> In version 5.5, the old Windows installer without JRE was discontinued and a specific installer, including the JRE, was provided for each operating system (Windows, Linux, and macOS). No languages are preinstalled for these new installers.

At launch, Audiveris checks the list of installed languages and issues a warning if this list is empty. In interactive mode, the following warning dialog is displayed:



Your collection of languages is empty!
Would you like to install some?

We can then:

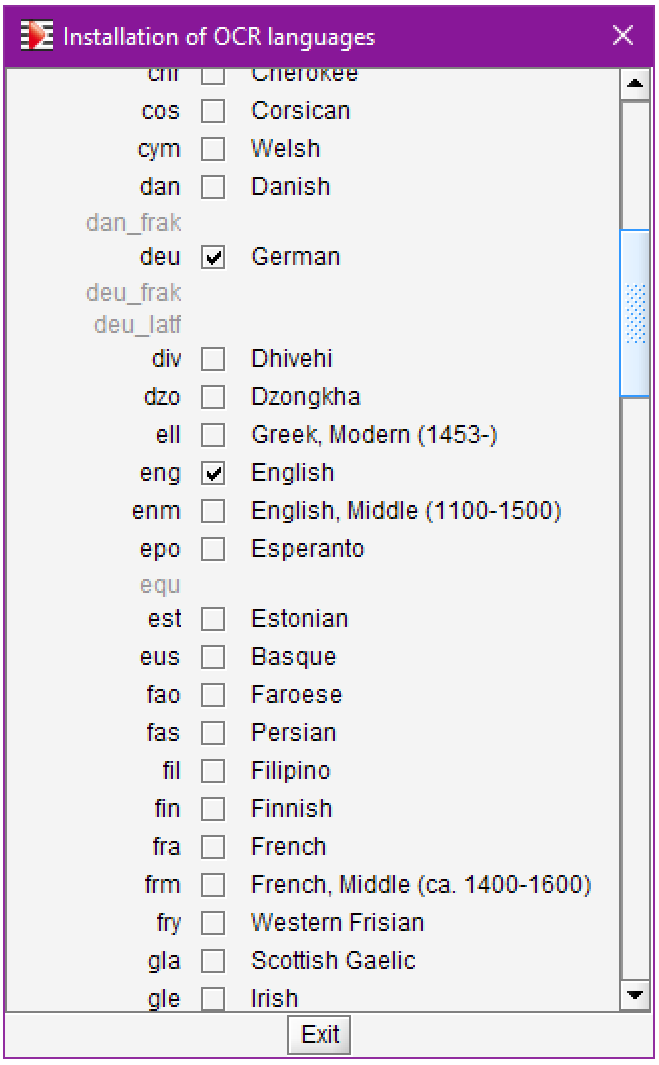
- proceed immediately with the installation of some languages;
- or simply postpone this action; in the meantime, we can still process the input scores, but the TEXTS step will be ignored.

Installing languages

Prior to version 5.4, the end user was left alone to manually download data files from the Tesseract web site into a local folder. This resulted in recurring errors.[1]

Starting from version 5.4, Audiveris offers a convenient way to install OCR languages *interactively* from within the application.

The preferred method is to add languages via the `Tools → Languages` pull-down menu.
Of course, *this feature requires Internet access*.
If OK, we get the dialog below:



chr	☐	Cherokee
cos	☐	Corsican
cym	☐	Welsh
dan	☐	Danish
dan_frak		
deu	☑	German
deu_frak		
deu_latf		
div	☐	Dhivehi
dzo	☐	Dzongkha
ell	☐	Greek, Modern (1453-)
eng	☑	English
enm	☐	English, Middle (1100-1500)
epo	☐	Esperanto
equ		
est	☐	Estonian
eus	☐	Basque
fao	☐	Faroese
fas	☐	Persian
fil	☐	Filipino
fin	☐	Finnish
fra	☐	French
frm	☐	French, Middle (ca. 1400-1600)
fry	☐	Western Frisian
gla	☐	Scottish Gaelic
gle	☐	Irish

Here, we can browse through the 100+ languages available on the GitHub Tesseract site, one line per language:

- The language ‘code’ (generally 3 letters) appears on the left
- The full name is on the right
- The checkbox in the middle indicates if the corresponding data file is installed locally
- A gray code, with no checkbox and no full name, can be ignored

To actually download a language, we simply check the related box. The corresponding data file is immediately downloaded to the local target folder and the checkbox is updated once the download is complete.

The local `tessdata` folder

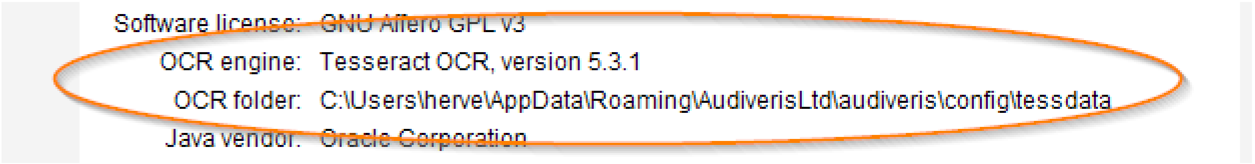
At launch, Audiveris tries to initialize the Tesseract library with a `tessdata` folder:

1. It first checks if the `TESSDATA_PREFIX` environment variable is defined and actually points to a directory. If so, this location is chosen.
   Note that setting this environment variable is not mandatory, it is meant merely to reuse an existing configuration.

2. Otherwise, Audiveris looks for a directory named `tessdata` under the OS-dependent Audiveris user config folder.
   If this directory does not exist, it is created on-the-fly.

NOTA: this target folder must be **writable** to actually install any language there.

The `About` dialog, launched from the `Help → About` pull-down menu, displays key information about the OCR engine version and the local OCR tessdata folder:



Software license: GNU Affero GPL v3
OCR engine: Tesseract OCR, version 5.3.1
OCR folder: C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\config\tessdata
Java vendor: Oracle Corporation

Specifying runtime languages

At runtime, we can specify which languages should be considered by the OCR software for the input image at hand.

This is done via a language specification string – a plus-separated list of language codes, like “fra+eng+ita”:

- The easiest way is to define this language specification interactively.
  Using the `Book → Set book parameters` menu, we can make specifications at the global level, book level and even individual sheet level.

- The default (global) specification is determined by the application constant `org.audiveris.omr.text.Language.defaultSpecification`, whose initial value is `eng`.
  So, we can also modify this default directly by changing the constant value:
  - either interactively (using the `Tools → Constants` menu)
  - or in batch (using something like `-constant org.audiveris.omr.text.Language.defaultSpecification=ita+eng`).

> NOTA: Specifying too many languages for a given book or sheet will slow down the recognition task and may increase the number of false recognitions.
> So, let us be as specific as possible.

Footnotes

[1] Most frequent confusion cases:

- Tesseract *program* vs. Tesseract *library* (linked by Audiveris)
- Tesseract `LSTM` model vs. `legacy` model (the one needed by Audiveris)
- Version of Tesseract language files to be downloaded (4.x)
- `TESSDATA_PREFIX` environment variable (pointing to the local tessdata folder) ↩

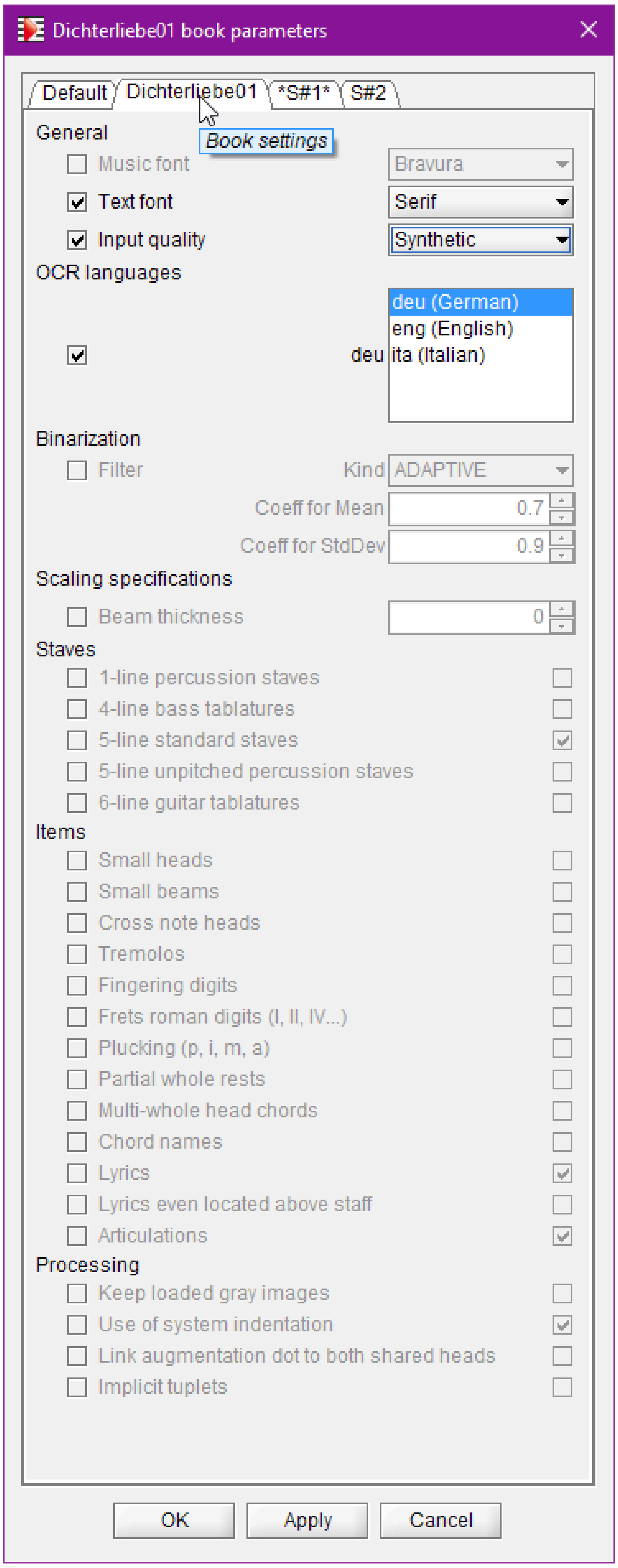
Book parameters

TABLE OF CONTENTS

Dialog

The pull-down menu `Book → Set book parameters` opens a dialog to review and modify high-level processing parameters.

The following example displays the parameters dialog for a book (`Dichterliebe01`) which contains two sheets:



Dichterliebe01 book parameters

General

☐	Music font	Bravura
☑	Text font	Serif
☑	Input quality	Synthetic

OCR languages

☑	deu	deu (German) eng (English) ita (Italian)

Binarization

☐ Filter	Kind	ADAPTIVE
	Coeff for Mean	0.7
	Coeff for StdDev	0.9

Scaling specifications

☐	Beam thickness	0

Staves

☐	1-line percussion staves	☐
☐	4-line bass tablatures	☐
☐	5-line standard staves	☑
☐	5-line unpitched percussion staves	☐
☐	6-line guitar tablatures	☐

Items

☐	Small heads	☐
☐	Small beams	☐
☐	Cross note heads	☐
☐	Tremolos	☐
☐	Fingering digits	☐
☐	Frets roman digits (I, II, IV...)	☐
☐	Plucking (p, i, m, a)	☐
☐	Partial whole rests	☐
☐	Multi-whole head chords	☐
☐	Chord names	☐
☐	Lyrics	☑
☐	Lyrics even located above staff	☐
☐	Articulations	☑

Processing

☐	Keep loaded gray images	☐
☐	Use of system indentation	☑
☐	Link augmentation dot to both shared heads	☐
☐	Implicit tuplets	☐

Scopes

The dialog is organized in several tabs to describe Default, Book and Sheet’s scopes respectively. In this example, the dialog provides 4 tabs, one for

Default, one for Dichterliebe01 book, and one for each sheet in the Dichterliebe01 book.

The same parameters are defined for each scope, and each scope by default inherits from the upper scopes in the hierarchy:

1. **Default** level: This is the information provided by default for all books. Any such global value is read from source, unless we have overridden it at default level.
2. **Book** level: We can override any default value for this book only, and it will apply transitively to all sheets in this book.
3. **Sheet** level: Finally, we can override any value for the specific sheet at hand.

To override a value for a given scope:

1. We first select the proper scope tab,
2. We then put a checkmark on the left side to indicate that we want to override the selected parameter. The line gets enabled, it changes from gray to black.
3. We define the new value, either by selecting in a list, or typing a number, or checking a boolean box, etc.

To cancel a value modification, we simply un-check the box on left side. The line then gets disabled, changing from black to gray, and it now displays the inherited value in lieu of the overriding value.

Lifecycle

All modifications apply only when either the `OK` button or the `Apply` button is pressed, which actually commits them. The `OK` button completes the dialog, while the `Apply` button keeps the dialog open.

- All the modified **default** values persist from one run of the application to the other (until they are modified again or reset to their factory values).

- All the modified **book/sheets** values persist in the book `.omr` project file.

Parameters

General

- **Music font**
  We can choose a specific music font between `Bravura`, `Finale Jazz` or `Jazz Perc`.
  This decision is especially important for head recognition which is based on a font template matching technique.
  See the specific Music Font section.

- **Text font**
  The text font has no impact on recognition, but can provide a more faithful output. We can adjust the text font between `Sans Serif`, `Serif` and `Finale Jazz Text`.

- **Input quality**
  This item allows adapting the OCR engine behavior to the expected quality of the input image between `synthetic`, `standard` and `poor`.

OCR languages

Define the specification of dominant languages for OCR'd text (note that we can select several languages)

Binarization

(this topic does not appear unless the SPECIFIC_ITEMS advanced topic has been selected)

We can select the kind of filter (`global` or `adaptive`) which gives the best results for the sheet image at hand.

We can also adjust the related numbers. Playing with the `global` threshold is easy, but modifying the parameters of the `adaptive` filter is not recommended. In fact, the default `adaptive` filter seems to give good results in all cases, therefore this parameter is kept only for potential use on a specific case.

Scaling specifications

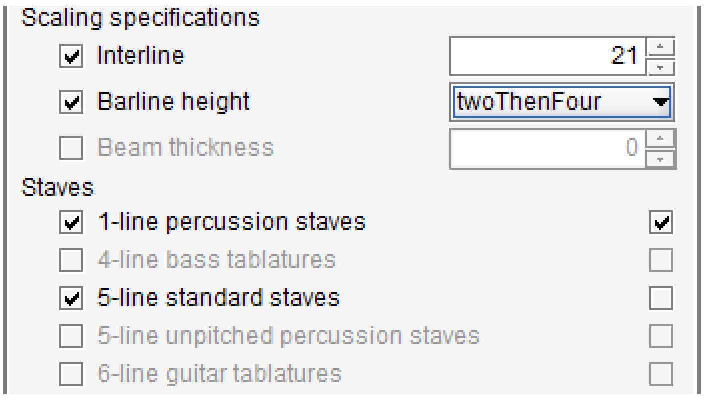
- **Interline** specification (expressed in number of pixels).
  This field appears when ***only*** 1-line staves are declared. In this case, the

engine has no way to measure an “interline value”, since there is simply *no* interline.

- **Barline height** specification (`four`, `twoThenFour`, `two`, `oneThenTwo`).
  This field appears when 1-line staves are declared. It defines the typical height for barlines in 1-line staves, expressed in number of interlines (or note head heights).
  - `four`: The height of every barline is 4 interlines (this is the usual case, as for 5-line standard staves)
  - `twoThenFour`: The height of the first barline (on the left side of the staff) is 2 interlines, and the following ones are 4 interlines.
  - `two`: The height of every barline is 2 interlines.
  - `oneThenTwo`: The height of the first barline (on the left side of the staff) is 1 interline, and the following ones are 2 interlines.
- **Beam thickness** specification (expressed in number of pixels).
  This field is always present, it is useful when there are too few beams for the engine to get a reliable measurement.

The scaling specifications can depend on staves declarations, as shown in the following picture.
Note in this example that:

- There are no multi-line staves, hence the need for interline specification
- There are 1-line staves, hence the need for barline height specification



Scaling specifications
☑ Interline 21
☑ Barline height twoThenFour
□ Beam thickness 0
Staves
☑ 1-line percussion staves ☑
□ 4-line bass tablatures □
☑ 5-line standard staves □
□ 5-line unpitched percussion staves □
□ 6-line guitar tablatures □

Staves

- Presence of 1-line percussion staves
- Presence of 4-line bass tablatures
- Presence of 5-line standard staves
- Presence of 5-line unpitched percussion staves
- Presence of 6-line guitar tablatures

Processing

In some cases, supporting a rather rare feature may imply collateral damage, the small note heads are an example of such a tricky feature. So it is safer to use them only when we have to.

- Keep loaded gray images
- Use of system indentation
- Potential presence of small note heads
- Potential presence of small beams
- Potential presence of cross note heads
- Potential presence of tremolos
- Potential presence of fingering digits
- Potential presence of frets roman digits (I, II, III, IV…)
- Potential presence of plucking (p, i, m, a)
- Potential presence of partial whole rests
- Potential presence of multi-whole head chords
- Potential presence of chord names
- Potential presence of lyrics
- Potential presence of lyrics even located above staff
- Potential presence of articulations
- Link augmentation dot to both shared heads
- Support for implicit tuplets

> A switch can disable a feature for the OMR automatic recognition, but in most cases the feature remains available for manual user actions.

Sheet scale

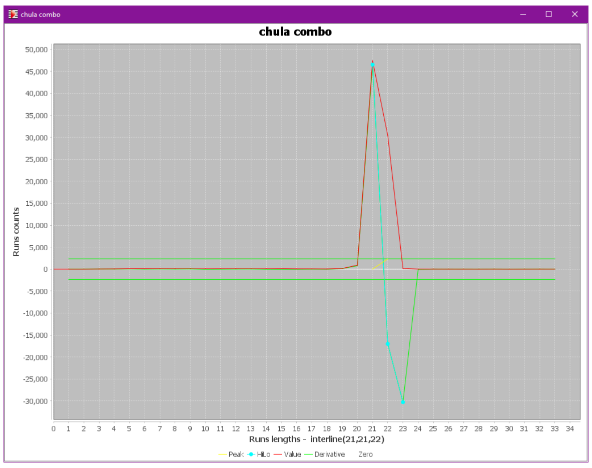
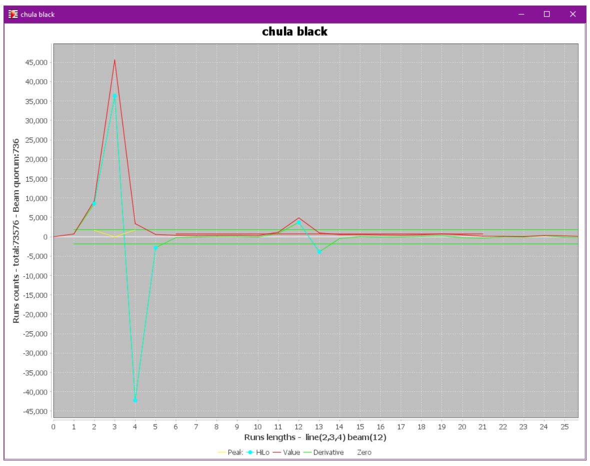
TABLE OF CONTENTS

General scaling

The behavior of steps like GRID (staves, etc), BEAMS and STEMS depends highly on the accuracy of scaling data estimated during:

- `SCALE` step: Staff line thickness, interline, small interline if any, beam thickness, small beam thickness if any.
- `STEM_SEEDS` step: Stem thickness.

Scaling data is inferred from histograms of vertical run lengths.
Here are two examples of such histograms:

Combo histogram	Black histogram
	

- The combo histogram focuses on the combined length of a black run and its following white run. This leads to the measurement of staff interline.
- The black histogram focuses on the length of vertical black run.
  - The main peak corresponds to the typical staff line thickness (3 pixels in this example).
  - If the sheet contains a significant population of beams, we can observe a second peak which corresponds to the typical beam thickness (12 pixels in this example).

Beam thickness

The Audiveris OMR engine, in its BEAMS step, makes a pretty good processing of beams – when it knows the typical beam thickness in the sheet at hand.

A problem arises if the sheet at hand contains only a couple of beams: The contribution of these beams to the global black histogram may be just too low to result in any reliable peak.

In this case, the OMR engine may pick up a wrong beam peak or detect no beam peak at all and simply make a guess on beam thickness (using say 1/2 of staff interline).
And without a precise thickness estimate, chances are the beams will not be correctly detected and processed in the sheet.

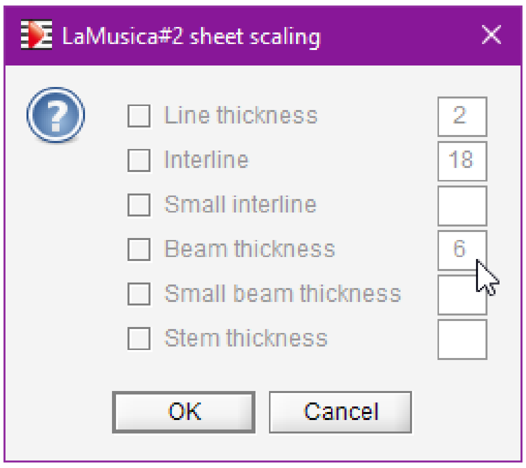
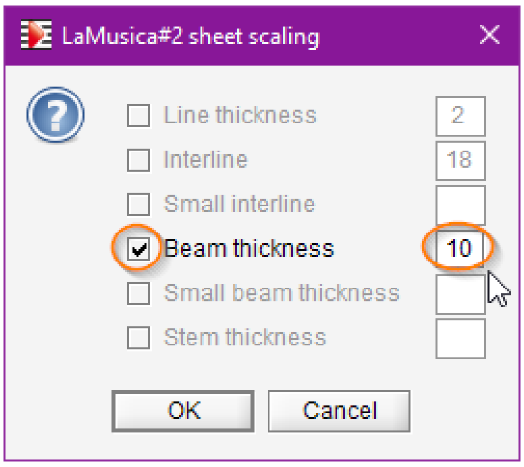
Manual beam correction

After the SCALE step, but before the BEAMS step, we can manually modify the beam thickness value as computed by the OMR engine.

We can do this using the `Sheet → Set scaling data` pull-down menu.

Initially, all rows appear in gray, and the data values, if any, are displayed on the right.

- To modify a data, we first select the row by checking the box on the left; the row turns black.
  We can then modify the data value.
- Deselecting a row resets the data to its initial value.
- Finally, pressing the `OK` button commits the modified values and closes the dialog.

Original scaling	Edited scaling
 LaMusica#2 sheet scaling ☐ Line thickness 2 ☐ Interline 18 ☐ Small interline ☐ Beam thickness 6 ☐ Small beam thickness ☐ Stem thickness 	 LaMusica#2 sheet scaling ☐ Line thickness 2 ☐ Interline 18 ☐ Small interline ☑ Beam thickness 10 ☐ Small beam thickness ☐ Stem thickness

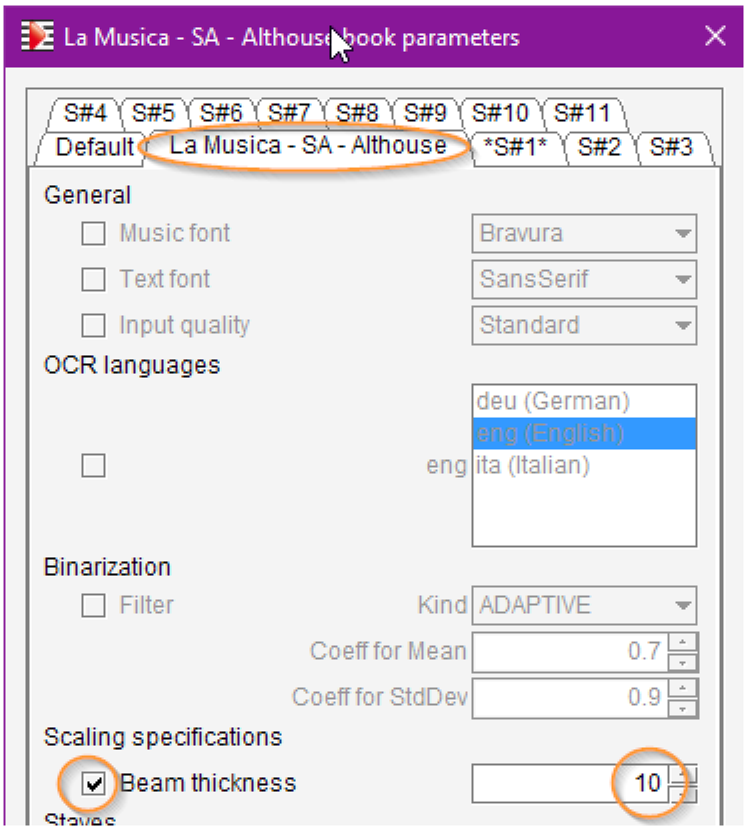
- Pros:
  - We directly tell the engine which beam thickness to use in this sheet.
- Cons:
  - Each time the processing of this sheet is restarted from zero, the correction is lost and the modified value must be re-entered.
  - We may have to perform the correction manually for each and every sheet in the containing book.

Upfront beam specification

A different approach is to provide upfront the OMR engine with a beam thickness specification.

Interactive mode

We can do this using the `Book → Set book parameters` pull-down menu:



La Musica - SA - Althouse book parameters
S#4
S#5
S#6
S#7
S#8
S#9
S#10
S#11
Default
La Musica - SA - Althouse
S#1
S#2
S#3

General

□ Music font Bravura

□ Text font SansSerif

□ Input quality Standard

OCR languages

□ eng deu (German) eng (English) ita (Italian)

Binarization

□ Filter Kind ADAPTIVE

Coeff for Mean 0.7

Coeff for StdDev 0.9

Scaling specifications

☑ Beam thickness 10

Staves

Here, we have chosen the whole book tab, selected beam specification and entered a thickness value as a number of pixels.

Since the specification is made at book level, it applies by default to all sheets in the book. This is generally a good strategy, since often all sheets in a book share the same scale and the same beam thickness.

If ever we need a different value for a given sheet, we can simply select the related sheet tab in the `Book parameters` dialog, and enter a specific value for this particular sheet.

Specification value `0` has a special meaning, which is: `no specification`. It can be used at book and/or sheet levels.

- Pros:
  - It's easy to set book-wide as well as sheet-specific specifications.
  - The specification(s) are stored in the `.omr` project file and survive any reload or any processing restart from scratch.
- Cons:
  - It is only a *specification*, to be used by any subsequent SCALE step. Hence, to be effectively applied, the SCALE step must be performed (or reperformed).

Batch mode

When running in batch on a brand new input, we can include this beam specification in the command line arguments, using an option like:

> -constant org.audiveris.omr.sheet.Scale.defaultBeamSpecification=10

As opposed to the interactive mode, this batch specification is *not* stored within the book project file.
And we must keep in mind the fact that, in batch, this beam thickness specification applies to *all* books processed by the current command entered on the command line interface.

Similarly, we can notice that there is no beam specification available in the `Default` tab of the `Book parameters` dialog. Otherwise this hard-coded pixel specification would apply for all books processed from now on…

Book portions

Handling a one-image score is just a matter of one book with one sheet in it. But handling cases of dozens or hundreds of score images may require more flexible approaches:

- Imagine we have scanned several images resulting in several separate image files. If these images belong to the same movement, we will need a way to gather all these pieces at some point in time, in order to export the result as one MusicXML score.
- Imagine on the contrary that we have a single huge input file that we would like to work on, chunk by chunk at our own pace, and export either the whole thing or just one portion at a time.
- Imagine we have already worked on a long book, transcribing and correcting most of it, just to discover that one input sheet would definitely require some image pre-processing or even a new scan.
  In such a case, we would appreciate the ability to insert or replace a few sheets in the book at hand while keeping the other sheets intact, especially the ones already transcribed.

In this chapter, we will discover that Audiveris has now departed from its initial approach of *“one input file - one book”*.

TABLE OF CONTENTS

How-to guides / Main features / Book portions / Sheet validity

Sheet validity

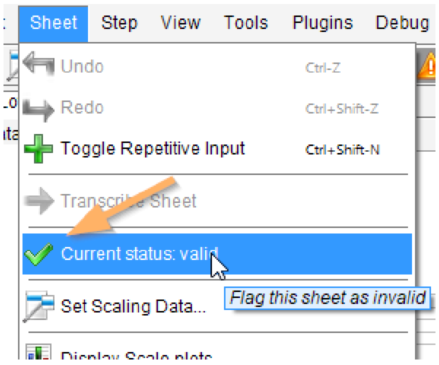
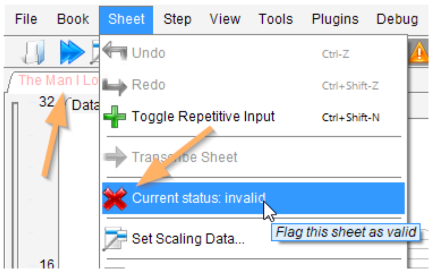
All images in an input file may not be score images: we can find illustration pages, blank pages, etc. Generally, the SCALE step is smart enough to detect that there is not enough “staff material” in a sheet and thus detect an invalid sheet from the OMR point of view.

When running in batch, the engine sets the *invalid* flag for the sheet. When in interactive mode, we are prompted to confirm whether the sheet is really invalid.

How to know if a given sheet has been set to valid or invalid?

In the `Sheet` pull-down menu, we can look at the `Current status` item:

- It indicates the sheet current validity status
- By pressing on the menu item, we toggle the sheet validity status.

Valid sheet	Invalid sheet
 Sheet Step View Tools Plugins Debug Undo Ctrl-Z Redo Ctrl+Shift-Z Toggle Repetitive Input Ctrl+Shift-N Transcribe Sheet Current status: valid Flag this sheet as invalid Set Scaling Data... Display Scale plots	 File Book Sheet Step View Tools Plugins Debug Undo Ctrl-Z Redo Ctrl+Shift-Z Toggle Repetitive Input Ctrl+Shift-N Transcribe Sheet Current status: invalid Flag this sheet as valid Set Scaling Data...

Also, the text on the sheet tab turns red for an invalid sheet.

We can decide to display or not the tabs for invalid sheets, via the menu item `View | Display invalid sheets`.

A sheet declared as `invalid` is no longer processed: no transcription, no export, no printing, nothing! Unless of course if we realize we’ve made a mistake and decide to set it back to `valid` after all!

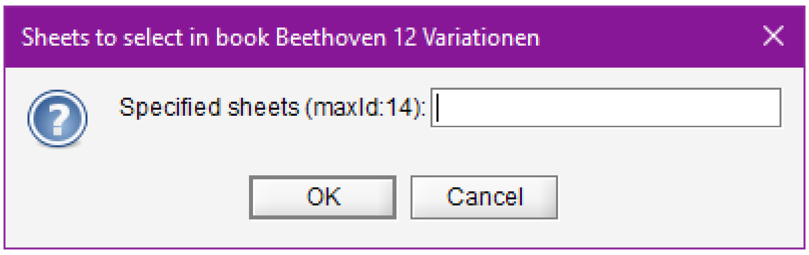
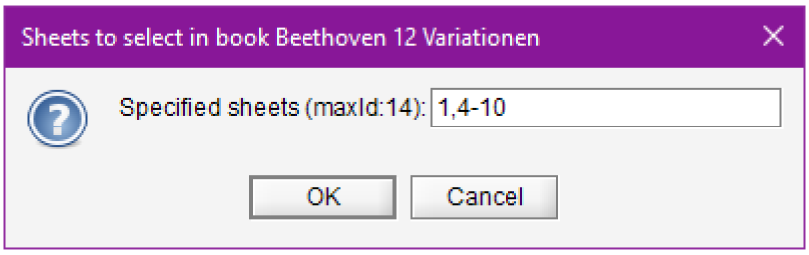
But validity is supposed to represent the real status of the sheet from the OMR perspective; we should not use it to include or exclude this sheet for some processing. A side effect of modifying the validity status of a sheet is to nullify all its OMR data!

Instead, we should use the sheet selection mechanism or the even more powerful Split and merge tool that can be accessed via the `Book → Split and merge` pull-down menu.

Sheet selection

An action launched at the sheet level processes only that sheet.
An action launched at the book level, processes by default all the (valid) sheets of the book. And this may be too much when what we want is to work only on a portion of the book (for example to print or listen to just a movement or a few selected sheets).

We can specify a sheet selection, via the pull-down menu `Book → Select sheets` which opens a selection dialog based on sheet IDs as shown below: (The highest sheet ID in the current book is recalled)

Selection	Meaning
 Sheets to select in book Beethoven 12 Variationen Specified sheets (maxId:14): 	Blank specification: **All** sheets are selected
 Sheets to select in book Beethoven 12 Variationen Specified sheets (maxId:14): 1,4-10 	Specification says: 1, then 4 through 10

Remarks:

- This is only a *specification*. To be really processed, each selected sheet must also be *valid*.
- A sheet specification must be written in a strict increasing order.
- Specifications “1,4-10” and “1,4,5,6,7,8,9,10” are equivalent.
- Since there is a gap in this example (sheets 2 and 3 are not selected), exporting the book to disk or plugin will result in at least two separate movements, one for sheet 1 and one for sheets 4-10.
- If the specification string is blank or null, all sheets are selected by default. [1]
- A sheet specification can also be provided in batch via the `-sheets` argument

on the command line interface.

- The latest sheet specification is persisted in the book `.omr` project file when specified via the GUI dialog but not when specified via the command line interface in batch.

Footnotes

[1] In Audiveris 5.2, a null specification used to mean *all sheets* while an empty specification used to mean *no sheets*.
Starting with Audiveris 5.3, for the sake of consistency with the new “Split and merge” feature, they now both mean the same thing: ***all sheets***. ↩

Split and merge

This feature is meant to be the Audiveris “Swiss Army Knife” to configure books, images and sheets.

TABLE OF CONTENTS

Concepts

The Split and merge feature relies on the concept of *“PlayList”* that the user can define, modify, save/reload, and finally use to build the resulting compound book.

In terms of structure:

- A PlayList is merely a sequence of excerpts to be processed one after the other.
- An Excerpt represents, not surprisingly, a selection of sheets in a given Container.
- A Container is a file, either a Book file or an image(s) Input file.

The PlayList can be defined conveniently via the interactive “Split and merge” dialog.
It can also be written from scratch via a plain text editor (see the paragraph about the PlayList format).

Either way, the PlayList can then be used interactively or in batch to produce a compound book according to the PlayList content.

Use cases

This section lists typical use cases from the end-user point of view.

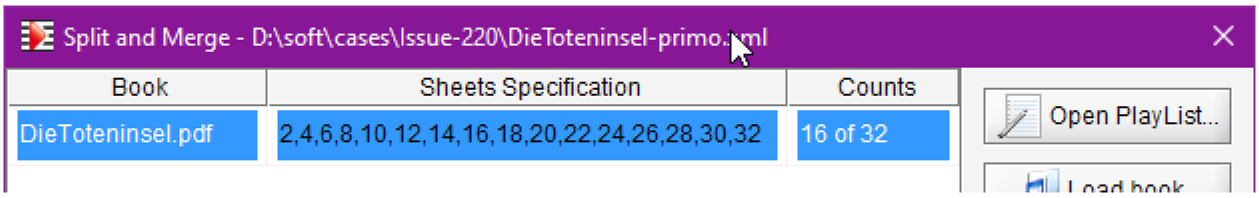
4-hands piano sheet

The request in issue #220 (Support for 4 hands piano sheet) was clearly stated:

> Some four hands piano sheets have a page/page layout, where left page is for left player and right page is for right player. The two pages are meant to be played at the same time.

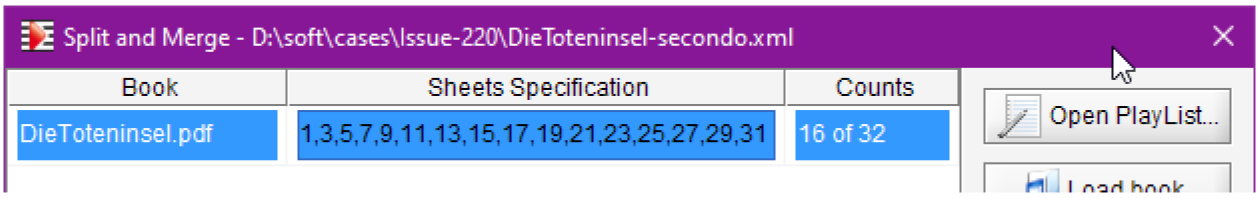
The provided input file for the case at hand is a 32-page pdf file, in which odd-numbered sheets are marked as “Secondo” and even-numbered sheets as “Primo”.

The easiest way to “split” the single input file into two separate books is to open two `Split and merge` dialogs and to edit them as follows:



Split and Merge - D:\soft\cases\Issue-220\DieToteninsel-primo.xml

Book	Sheets Specification	Counts
DieToteninsel.pdf	2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32	16 of 32



Split and Merge - D:\soft\cases\Issue-220\DieToteninsel-secondo.xml

Book	Sheets Specification	Counts
DieToteninsel.pdf	1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31	16 of 32

Then pressing the `Action` button in each dialog builds the corresponding compound books `DieToteninsel-primo.omr` and `DieToteninsel-secondo.omr` respectively.

Compound book

The case is described in issue #129 (Allow to load multiple images into one book) where a paper score is scanned page after page into separate image files.

1. With a PlayList referencing the image files in proper order, we can build a compound **book of images**.

2. We can also start processing each input file separately and perhaps try some image improvement techniques.
   And when each 1-sheet book is correctly processed, we can combine all the separate books into a compound **book of books**.

3. And more generally, we can have references to book files and references to image files in the same PlayList, resulting in a **mixed compound**.

In all these cases, the engine will update the sheets where needed and process the inter-sheet dependencies (slurs, time-signatures, measure IDs) before building the resulting score(s).

Sheet replacement

We have seen that sheet addition/removal can be done at any time, regardless of the transcription process of any sheet.

If, within a long book, a sheet needs some rework (such as a new input), then it’s simple to remove this sheet and insert a new version at the same position.

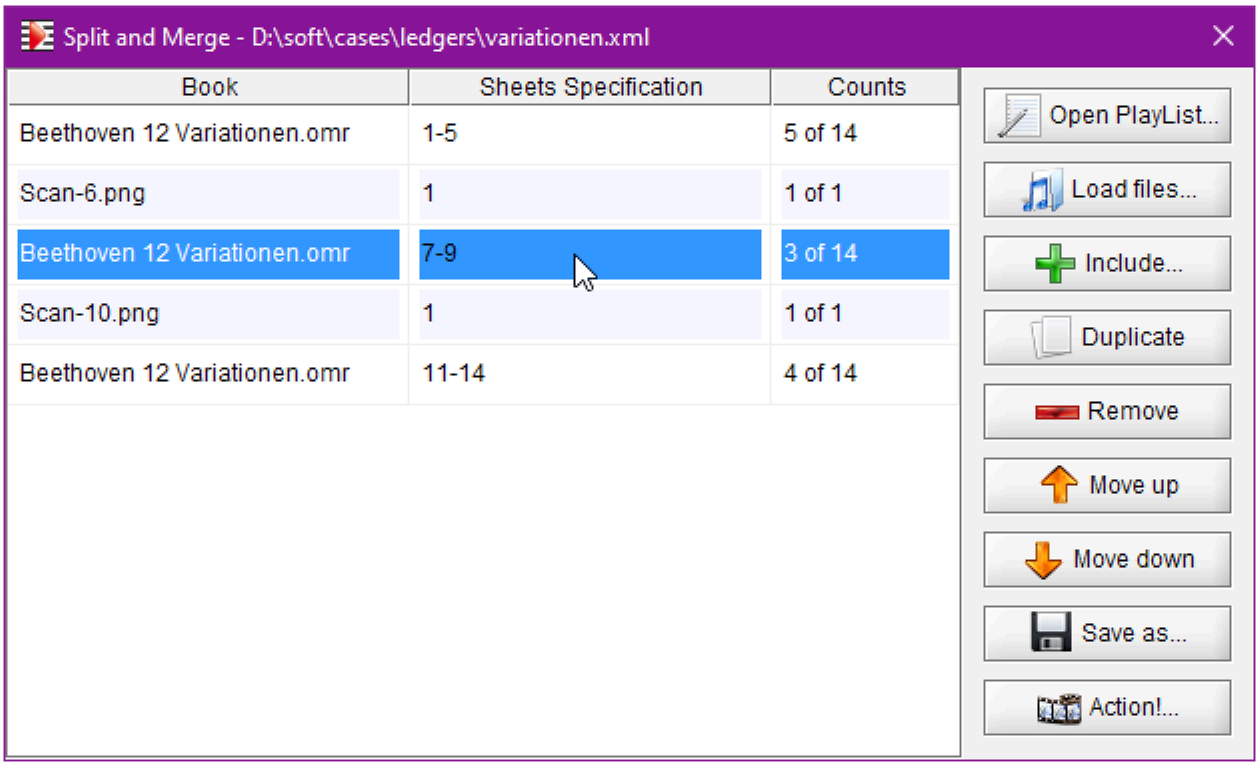
For example, let’s assume we have:

- `Beethoven 12 Variationen.omr`: a large book of 14 sheets, more or less well transcribed.

Let’s assume we have come to the conclusion that some input images should be replaced by other images of better quality:

- `Scan-6.png`: to replace sheet #6 of the book above
- `Scan-10.png`: to replace sheet #10 of the book above

Then we could use a PlayList as defined below:



Split and Merge - D:\soft\cases\ledgers\variationen.xml

Book	Sheets Specification	Counts
Beethoven 12 Variationen.omr	1-5	5 of 14
Scan-6.png	1	1 of 1
Beethoven 12 Variationen.omr	7-9	3 of 14
Scan-10.png	1	1 of 1
Beethoven 12 Variationen.omr	11-14	4 of 14

In this example, we are using references to `.png` image files. We could as well prefer references to `.omr` book files, containing the same sheets perhaps partly transcribed.

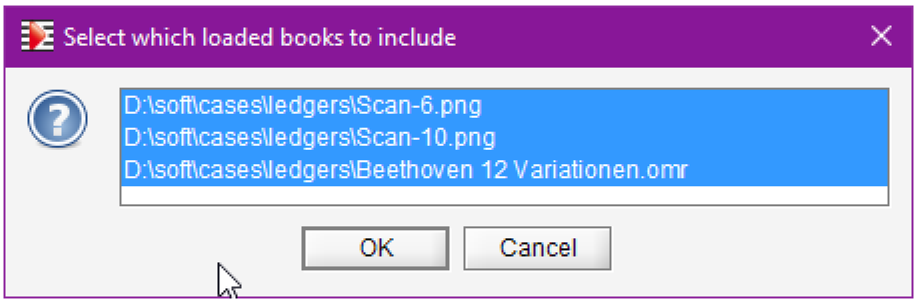
Dialog

It is accessed via the `Book → Split and merge` menu item.

The dialog large table, meant for the underlying PlayLists, is initially empty.

Populating the PlayList

- If we have a PlayList available somewhere, we can open it via the `Open PlayList...` button in the upper right corner of the dialog.
  This will add the content of the opened PlayList to the dialog.
- If we already have Books or Image files loaded in the Audiveris main window, we can immediately select and include a few of them into the PlayList.

  Select which loaded books to include

  D:\soft\cases\ledgers\Scan-6.png
  D:\soft\cases\ledgers\Scan-10.png
  D:\soft\cases\ledgers\Beethoven 12 Variationen.omr

- Via a Drag ‘n Drop from a file explorer, we can drop one or several Book or

Image files onto the dialog, and each will lead to the addition of a corresponding excerpt at a precise location in the PlayList.

- Similarly, via the `Load files` button, we can load one or several Book or Image files and their corresponding excerpts will be appended at the bottom of the PlayList.

Legend

- There is one row per excerpt, composed of the container name, the sheets specification string and the resulting count of sheets as selected in this excerpt.
- A container name ends with `.omr` extension to denote a Book, and with another extension (like `.png`, `pdf`, etc) to denote an image(s) file. The latter case is in reality a book created on-the-fly just to handle the image file.
- The sheets specification is an editable field. By default, all sheets are selected but, if the originating book already had a sheets selection defined, this definition would be copied over to the excerpt field. – see the previous section on Sheets Selection in a book.
- Note that the same container can appear on several rows of the PlayList, with different sheets specifications – see the example of Sheet replacement above.

Editing the PlayList

- We have seen that the sheets specification of any excerpt can be manually edited.
- We can add/include and remove excerpts.
- We can move a selected excerpt up and down within the PlayList.
- The `Duplicate` button duplicates the selected excerpt, including its current specification.
  It’s a convenient way to insert another excerpt of the same container in the PlayList: we can simply duplicate a line, move it where desired, and modify its sheets specification.

When satisfied with the PlayList, we can save it for future reload or use it immediately as follows.

Creating the compound book

The `Action!` button creates the compound book according to the PlayList current content.

The newly created Book is a full-fledged book that exists on its own.

It can be used from now on, just like any other book:

- whole book transcription,
- individual sheet transcription,
- interactive validation / correction,
- export in MusicXml, etc.
- it can even be used as a source for another PlayList.

The created book begins its life in some status, depending on the status of each of its sheets.

- it has simply been *created* by the `Action!` button,
- its full *transcription* will generally require some additional actions, like pressing the `Transcribe` button at book level.

> **IMPORTANT**
> We must keep in mind that this compound book is a brand new **stand-alone** book, kept separate from its original “parts”. In other words:
>
> - any further work made on this compound book will not impact the *original* “parts”,
> - any further work made on any of the *original* “parts” will not impact the compound book.

PlayList format

A PlayList can be created or saved as a plain XML file,

- with `<play-list>` as its root element,
- composed of a sequence of `<excerpt>` elements, each of which containing
    - the mandatory `<path>` element to a container file
    - and a possible `sheets-selection` element.

Here is the content of the `variationen.xml` example file, where the populated PlayList has been saved:

```
<?xml version="1.0" ?>
<play-list>
    <excerpt>
        <path>D:\soft\cases\ledgers\Beethoven 12 Variationen.omr</path>
        <sheets-selection>1-5</sheets-selection>
    </excerpt>
    <excerpt>
        <path>D:\soft\cases\ledgers\Scan-6.png</path>
        <sheets-selection>1</sheets-selection>
    </excerpt>
    <excerpt>
        <path>D:\soft\cases\ledgers\Beethoven 12 Variationen.omr</path>
        <sheets-selection>7-9</sheets-selection>
    </excerpt>
    <excerpt>
        <path>D:\soft\cases\ledgers\Scan-10.png</path>
        <sheets-selection>1</sheets-selection>
    </excerpt>
    <excerpt>
        <path>D:\soft\cases\ledgers\Beethoven 12 Variationen.omr</path>
        <sheets-selection>11-14</sheets-selection>
    </excerpt>
</play-list>
```

CLI option

There is a new option available on command line interface: `-playlist <foobar.xml>` which allows loading an `.xml` file as a PlayList, provided that file content is compliant with the PlayList format described above.

Then, according to the current mode (batch or interactive), the actions differ slightly:

- In *batch* mode, the PlayList is immediately used to build, next to the provided `foobar.xml` file, the resulting compound book `foobar.omr` file.
- In *interactive* mode, a `Split and merge` dialog is populated with this PlayList and displayed to the user for further action, such as review / edit / build compound.

User editing

In the transcription process made by the OMR engine, some elements may not be recognized correctly for various reasons, such as poor image quality or over-crowded score, to name a few.

There is a continuous effort going on to improve the OMR engine but on many examples a completely error free transcription is just out of reach. It is thus globally more efficient to complement the OMR engine with a convenient graphical editor so that the end-user can easily fix most of the OMR errors.

While Audiveris 5.1 release provided a basic editor limited to the drag & drop of fixed-size symbols, the 5.2 release added a comprehensive editor to modify nearly any kind of symbol.

The Audiveris editor does not try to compete with music editors like MuseScore, Finale or Sibelius. Those are sophisticated high-level editors meant for composers, arrangers or publishers who need a tool to create, arrange, transpose music, etc.

Instead, Audiveris is focused on the OMR process, which attempts to transcribe an existing score image to its symbolic representation. In particular, it tries to remain as close as possible to the original layout, including any image skew or warping.

Tutorial videos

As an introduction, you can watch Audiveris 5.1 user editor in action thanks to Baruch Hoffart who recorded this tutorial video.

For those who understand French, we highly recommend the YouTube channel run by Dominique Verriere. It is an impressive collection of detailed tutorials about Audiveris and MuseScore. Here is a direct link to his Audiveris videos.

Documentation content

1. UI foundations: Editable OMR data, how to select and inspect it.
2. UI tools: How to modify OMR data, beginning with the general tools to add/

edit/remove both Inters and Relations.
Then more ad-hoc tools are presented to address specific cases.

3. UI examples: Complete editing sessions on representative input scores.

The best way to learn an editor is certainly by practicing rather than studying a long documentation.
So, you could start by reading **Foundations** and then discover **Examples**.
Then, you could go back to the **Tools** of interest for you, at your own pace.

UI foundations

This chapter introduces the key components the end user can interact with, be they simple entities like glyphs, inters or relations or more complex containers like systems, measures and slots.

It focuses on their manual selection and on the most convenient ways to inspect their configuration.

TABLE OF CONTENTS

Principles

For Audiveris, music is something very graphical, a 2D view of music symbols (staff lines, bar lines, note heads, stems, flags, beams, alterations, …) with some “geometrical” relations between them (a note head and its stem, an alteration sign and the altered head, a lyric item and its related chord, a slur and its embraced heads, …).

The pixels of the score image are displayed in the background and serve as guidelines for the interactive user. Some pixels may have been gathered into **Glyph** instances by the OMR engine, but this “*segmentation*”, as it is called, is not always relevant, especially on poor quality images.

The entities that really matter for the final music content are the `Inter` (interpretation) instances and the **Relation** instances that formalize a relationship between two `Inter` instances, resulting in the **SIG** (Symbol Interpretation Graph) of each system.
If you are still unfamiliar with these notions of pixel / glyph / inter / relation / sig, please have a look at the dedicated Main concepts section.

We can think of Audiveris OMR as a process that takes an image as input and produces as output a structured collection of music symbols.

- The OMR *engine* is one actor, which moves forward from step to step, creating a bunch of entities.
- The OMR *user* is another actor, who can interactively modify some entities, and decide how to drive the engine.

The resulting objects are divided into two categories:

1. **Containers**. They are built at the beginning of the sheet processing and define the global structure of the sheet musical content, organized in a hierarchical structure of containers:
    - *staff lines*, gathered in …
    - *staves*, gathered in …
    - *parts*, gathered in …
    - *systems*, gathered in …
    - *pages*, gathered in …
    - *scores*

Since the 5.2 release, the user can partly modify this hierarchy, by splitting and merging systems, parts and measures.

2. **Inters**. They represent candidate interpretations, formalized by `Inter` instances, handled within each system by a graph (`SIG`) of `Inter` vertices linked by `Relation` edges. Within each system SIG, it’s the struggle for life, since only the strongest inters survive in the end.
   There is no structure within the `Inter` instances of a SIG, just `Relation` instances possibly set between a pair of `Inter` instances.
   Since the 5.2 release, all these `Inter` and `Relation` instances can be modified interactively.

   `Inter` objects are everything except the static containers mentioned above:

   - Barline, ledger, clef, key signature, time signature, chord, head, alteration, augmentation dot, stem, beam, rest, flag, slur, sentence, word, etc… all are examples of `Inter` ‘s.
   - An *InterEnsemble* is an `Inter` composed of other `Inter` instances. This composition is formalized by a `Containment` relation between the ensemble inter and each of the contained `Inter` instances.
     These ensembles can be:
     - A Chord which contains one or several note heads or one rest.
     - A Sentence which contains words. – A LyricLine is a special kind of Sentence, composed of LyricItem’s –
     - A Key which contains key alter signs.
     - A TimePair which contains a numerator number and a denominator number.
     - A BeamGroup which contains several parallel beams.
     - A StaffBarline which is a horizontal sequence of simple Barlines.

Measures are hybrid, since they are not Inters but depend on barlines which are Inters.

Beside `Inter` and `Relation` entities, the glyphs, which are just sets of foreground pixels, can be used to create new Inters. Hence, Glyph instances are also entities that the user can select and transcribe to `Inter` entities.

The Audiveris editor has been designed to work on the SIG data model, that is essentially to play directly with `Inter` and `Relation` instances.

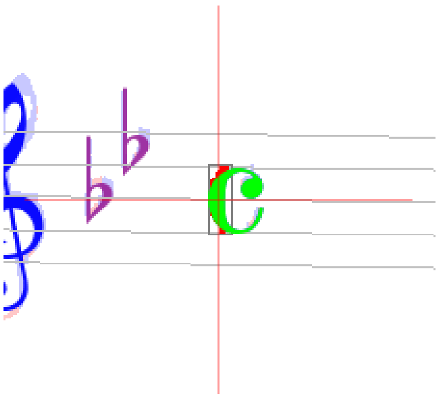
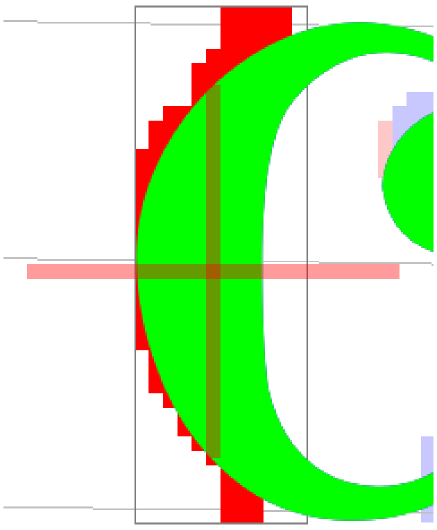
Selection

TABLE OF CONTENTS

Current location

Selection is driven by the current location we are pointing at in the score image.

The location point is displayed as a cross made of 2 red segments, one horizontal and one vertical. The length of each segment does not vary with the zoom ratio, but the segment thickness is always 1 pixel of the original image.

Zoom: 1	Zoom: 8
	

To define the current **point**, we either:

- Press down the left or the right mouse button at the desired location.
- Or, if the Pixel-Board is displayed, enter numeric values in the X and Y fields.

To define a current **rectangle** – which also defines the current point at the rectangle center –, we either:

- Press down the left mouse button at a desired rectangle corner location, and keep the `SHIFT` key pressed while dragging the mouse.
- Or, if the Pixel-Board is displayed, type numeric values in the X, Y, Width and Height fields.

To precisely **shift** the current location:

- While keeping the `ALT` key pressed (`OPTION` key on macOS), we use one of the 4 arrow keys to move the location (point or rectangle) one pixel at a time.

Entity selection

Selection modes

There are 3 selections modes available:

1. *glyph+inter* based (the default)
2. *inter*-only based

3. *section*-only based.

To switch from one mode to another, we use the toggle menu item `View → Switch selections`, click on the related toolbar icon or press the `F11` function key.

Selection means

The mouse-based selection works as expected, pointing to glyph entities, inter entities, section entities, according to the current selection mode.

- **Pointing to an entity**:
  - A *left*-button mouse click grabs the entities that contain the location point.
    If several entities are grabbed, only one is chosen to start the new selection.
    - For glyphs, it’s the smallest one.
    - For inters, it’s the member before the ensemble.
  - A *right*-button mouse click opens the pop-up menu:
    - with just the pointed entity if the previous selection was empty,
    - with the previous selection if the previous selection was not empty.
- **Using a lasso**: Pressing the `SHIFT` key while dragging with the mouse left-button defines a rectangular area.
  All entities *fully contained* in this area start a new selection.
- **Adding an entity**: A left click on an entity while pressing the `Ctrl` key (`Command` key for macOS) adds this entity to the current selection, whether the selection was done via point or lasso.
- **Naming an entity**: Entering the integer ID of a glyph in the Glyph-Board spinner or the ID of an `Inter` in the `Inter`-Board spinner selects this entity.
- **Pointing outside of any entity**:
  - A left click clears both glyphs and inters selections.
  - A right click opens the pop-up menu (with the entities selected so far).
- **Choosing an entity in the selection list**: The contextual pop-up menu, when entities have been selected, offers in the `Glyphs` and `Inters` sub-menus the list of selected entities. Simply moving the mouse onto the entity of choice will select this entity.
-

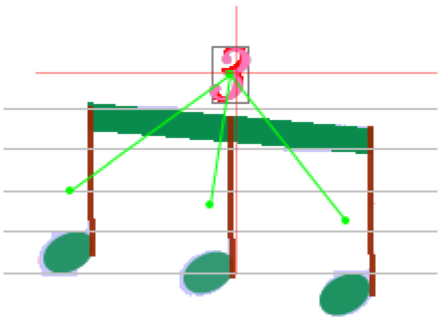
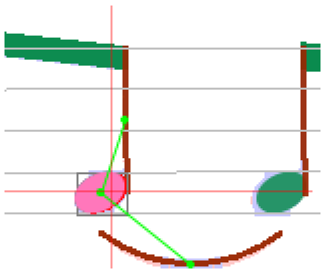
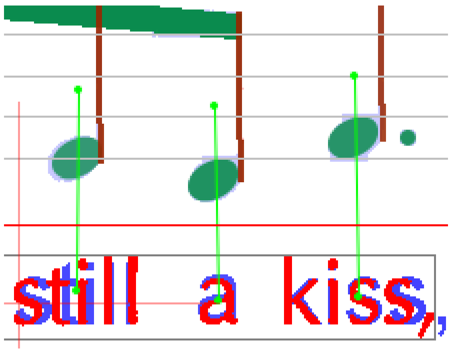
Choosing a relation in the relations list: Via the contextual pop-up menu, moving to a selected inter displays a sub-menu with all the relations the inter is part of. Selecting a relation offers to dump or delete this relation.

- **Double-click**: A double-click with the left mouse button on an `Inter` (not a Glyph) selects that `Inter` and puts it in `Editing mode` (see `Inter` editing).

A selected inter may display links to its related inter entities. The links appear as short straight lines in light green color – together with the relation name if the zoom is high enough.

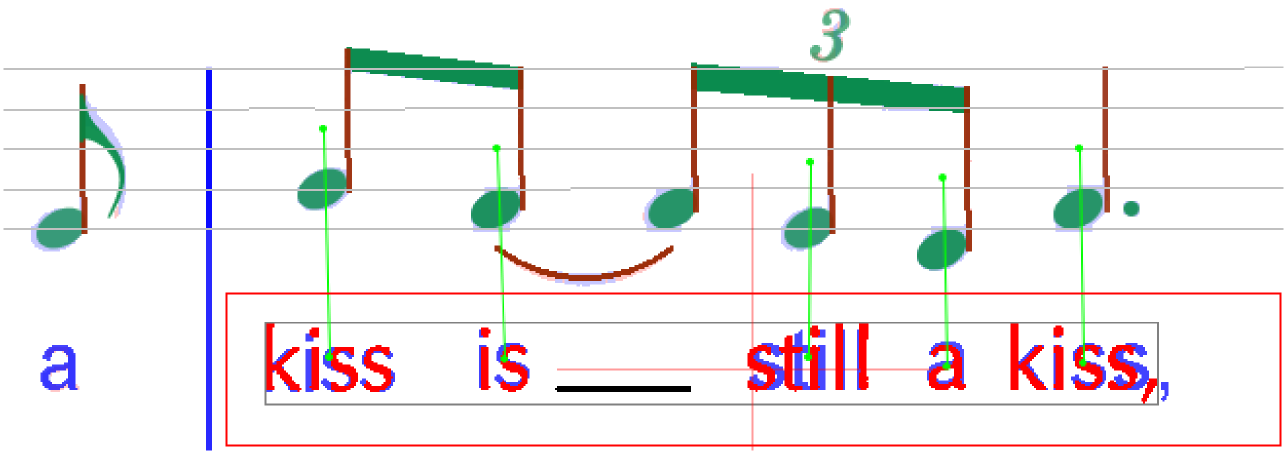
The images below depict:

1. A tuplet glyph linked to its 3 embraced chords
2. A note head connected to a stem and to a slur
3. Lyrics text with links to the related chords

Tuplet	Head	Lyrics
		

Multi-selection

A left-click in an entity area selects this entity (and deselects the entity previously selected if any).



To select several entities:

-

Either, by dragging the mouse while keeping the `SHIFT` key pressed, we use a rectangular “lasso” to grab all the entities whose bounds are **fully contained** by the lasso rectangle.

- Or, we select each entity, one after the other, keeping the `Ctrl` key pressed. If we have selected an item that we don’t want to keep, we simply click on it a second time and it will be de-selected. This can especially be useful when having selected several items using the “lasso” to then de-select items that are not wanted.

A red rectangle always shows our selection frame. When in *glyph+inter* mode, an additional black rectangle shows the bounding box of a compound glyph built from all the selected glyphs.

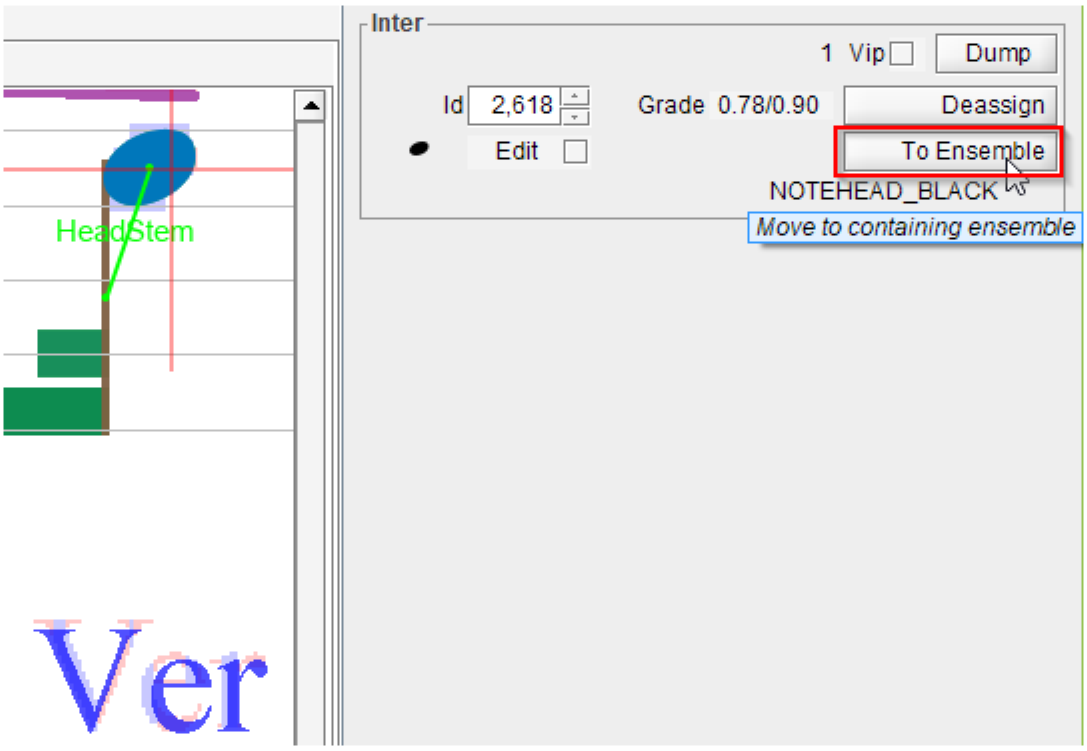
Moreover, an entity changes color when it gets selected (a glyph turns red, an inter turns to nearly-reverse color, a section turns white).

Member vs. ensemble

Selecting an entity by pointing at it can be ambiguous.

This can be the case when two or more **Glyphs** overlap each other. When we point at an overlapping region, the software will pick up only the smallest glyph.

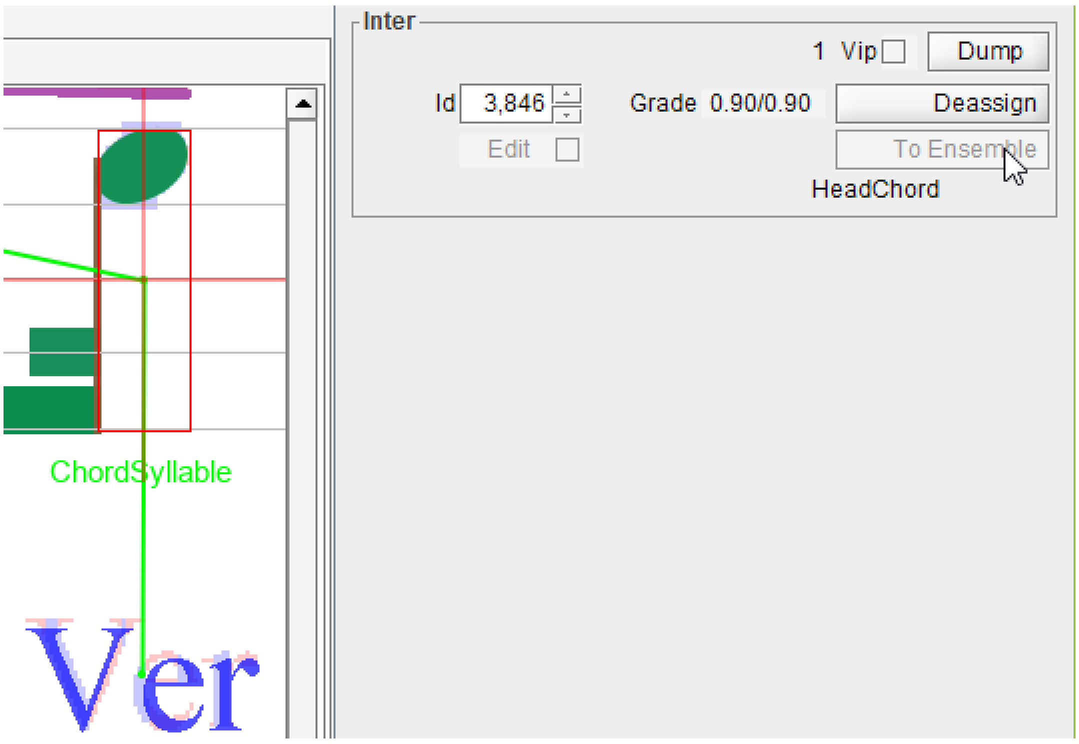
This is by definition always the case when pointing at an `Inter` which is a member of an `Inter` **ensemble**, for example a note and its containing chord. In this case, the software will choose the member over the ensemble, knowing that we can later switch from selected member to selected ensemble in the `Inter`-board (and in the Inters pop-up menu).
For example:



Id 2,618 Grade 0.78/0.90

NOTEHEAD_BLACK

Here, we have pointed on a note head which is part of a head chord. We can note the (member) head inter is selected over the (ensemble) chord inter, the involved “*HeadStem*” relation is visible between head and stem, the `Inter`-board focuses on the head inter, with the `To Ensemble` button enabled.



Id 3,846 Grade 0.90/0.90

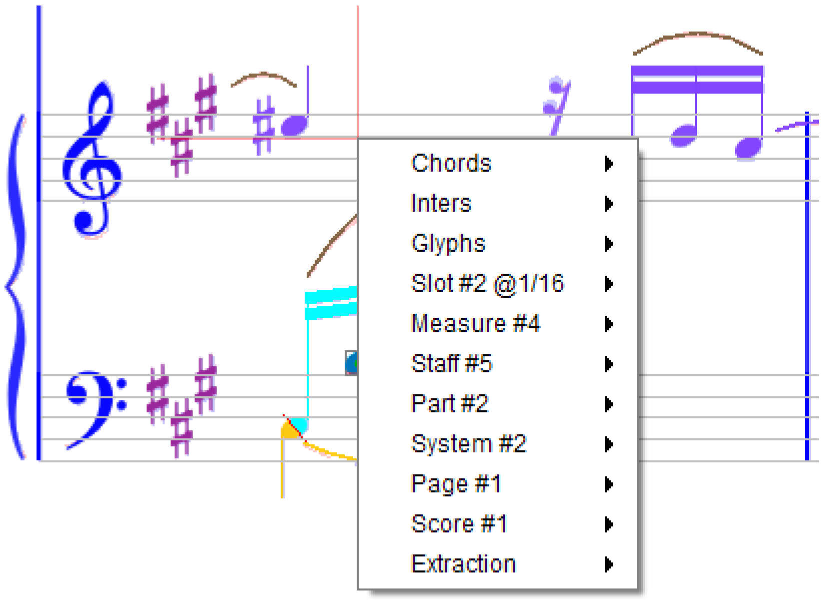
HeadChord

Here, we have pressed the `To Ensemble` button, and the focus is now on the head chord, with the “*ChordSyllable*” relation to the lyric item “Ver” below.

Container selection

When clicking with the right mouse button on any location of the sheet image, the sheet contextual menu appears.

This pop-up displays contextual information – such as the selected glyphs, inters, etc, if any – together with the containers being pointed at.



For example, the pop-up above allows to access – beside chords, inters and glyphs – the containers related to current mouse location:

- Time slot,
- Measure,
- Staff,
- Part,
- System,
- Page in sheet,
- Score,
- Whole image.

Then, each sub-menu leads to possible actions related to the container at hand.

Inspection

In a perfectly transcribed score, all the items would be correctly recognized and every chord in every voice in every measure would start at the right time and span the right duration.

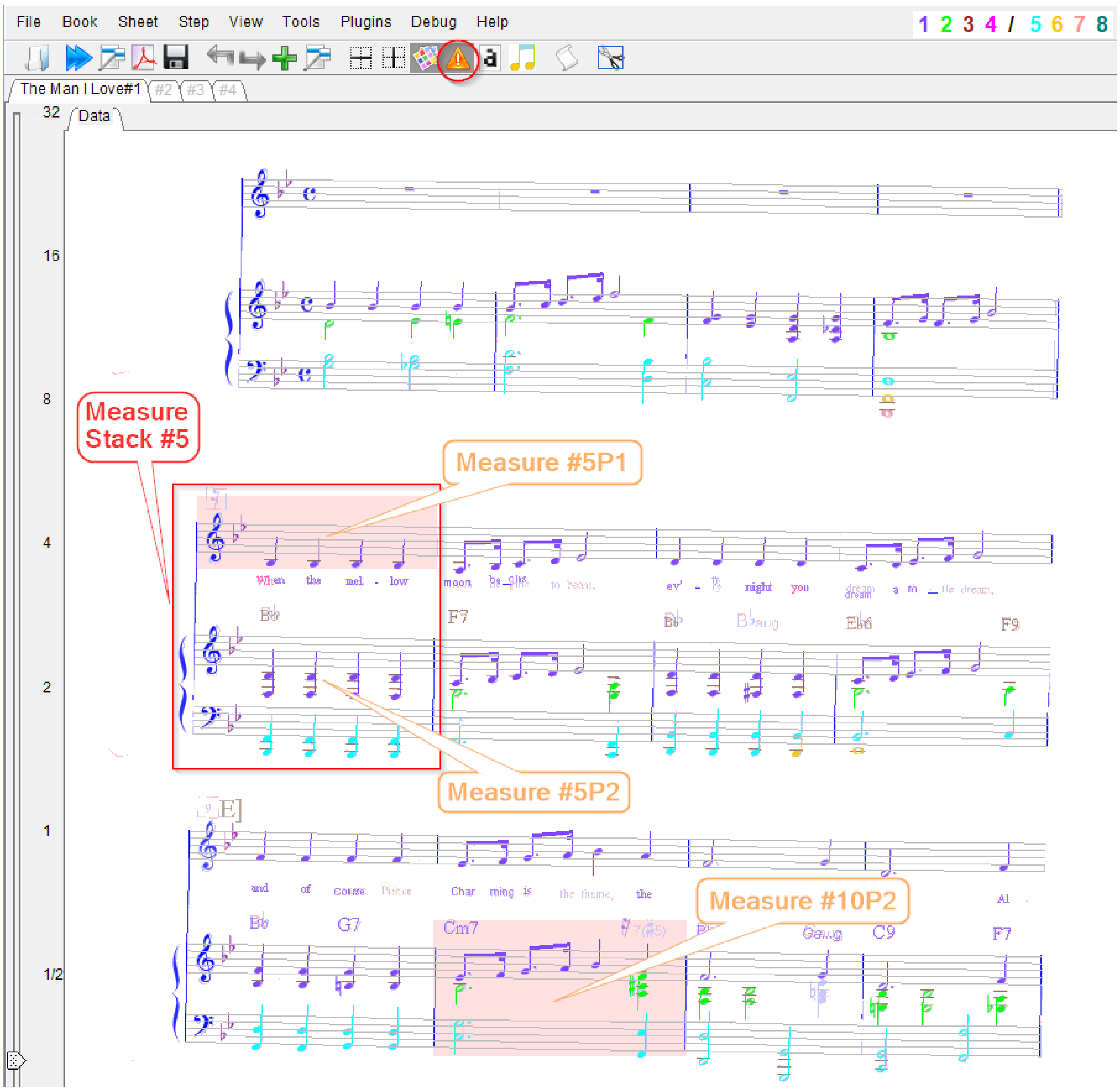
This beautiful mechanism can seize up for various reasons, large or small, and lead to abnormal results.

The purpose of this inspection chapter is to present various means for detecting such abnormal results and for discovering their root cause(s).

TABLE OF CONTENTS

Measure background

A pink measure background is the most obvious sign, meant to call user attention to a measure, as in the following page view which displays two measures (#5P1 and #10P2) with a pink background and the others with standard white background:
[Unless we have disabled this feature via the menu item `View → Show score errors` or the `F9` function key or the related toolbar icon]



Even though the rhythm is defined at the *measure stack* level (denoted by the system-high red rectangle in the picture above), abnormal rhythm is displayed down to the *measure* level (the part-high rectangle with pink background).

The measure background is defined:

- Horizontally by measure left and right limits,
- Vertically as the bounding box of all staves and chords involved in measure rhythm.

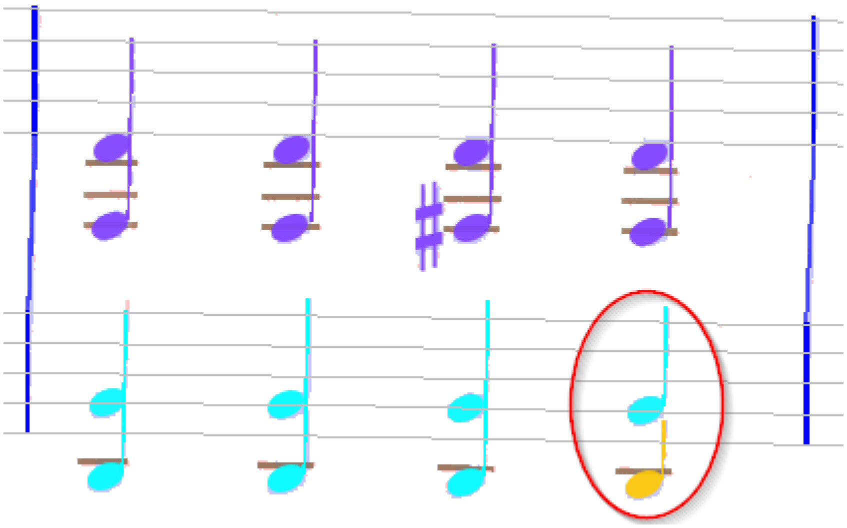
A measure is detected as *abnormal* – and thus displayed in pink – if, for at least one of its voices:

- The voice **starting time cannot be determined**, or
- The voice **ending time exceeds the measure expected duration** – as defined by the current time signature

Note that, conversely, a measure not detected as abnormal – and thus displayed with the standard white background – may still exhibit errors that the

OMR engine could not reliably detect. We’ll see examples of these cases that today only the human reader can detect and fix.

Colorized voices



The interest in choosing to colorize voices (see Voice colors) is to visually check the content of every voice.

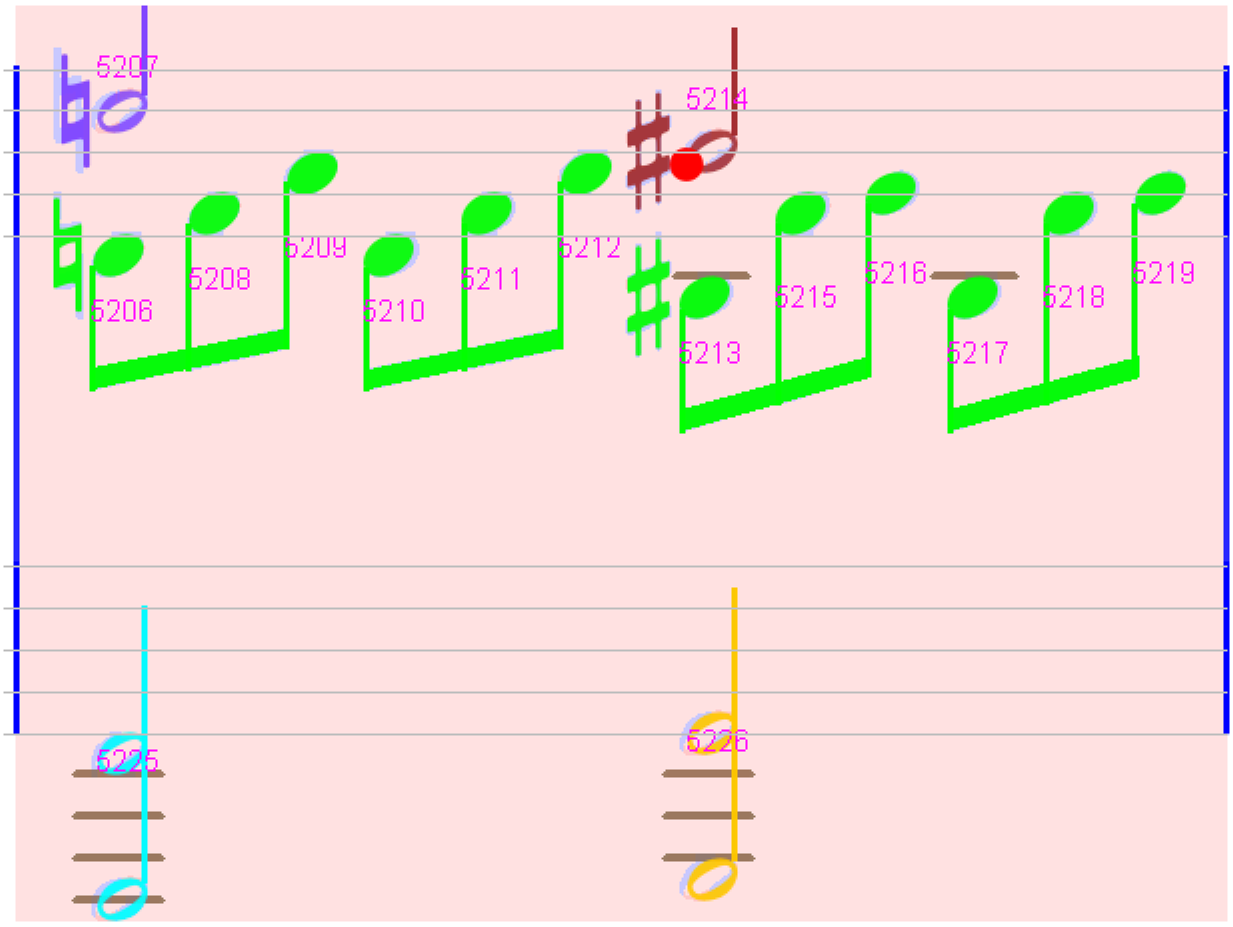
In the example above, voice 6 which appears in the bottom right corner of the measure is really suspicious (although the measure is not flagged as *abnormal*).

Jumbo mode

Very small glyphs located near note heads can be mistaken for augmentation dots by the OMR engine, resulting in erroneous durations and abnormal measures. Such small glyphs are often the result of poor segmentation and in many cases we cannot easily detect them by visual inspection.

Since the 5.4 release, there is a view option called the “Jumbo” mode, selectable via the `View → Show jumbo inters` menu item (or the `F7` funtion key).

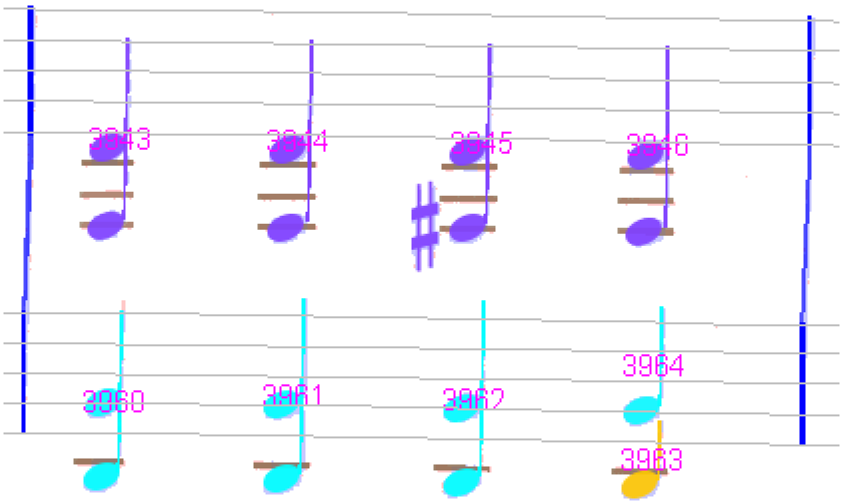
When this option is set, all augmentation dots in the current sheet are displayed as big red dots, so they can really stand out:



Chords IDs

By default, chord IDs are not displayed in sheet view because they look too invasive. To set them on, we use the pull-down menu `View → Show chord IDs`.

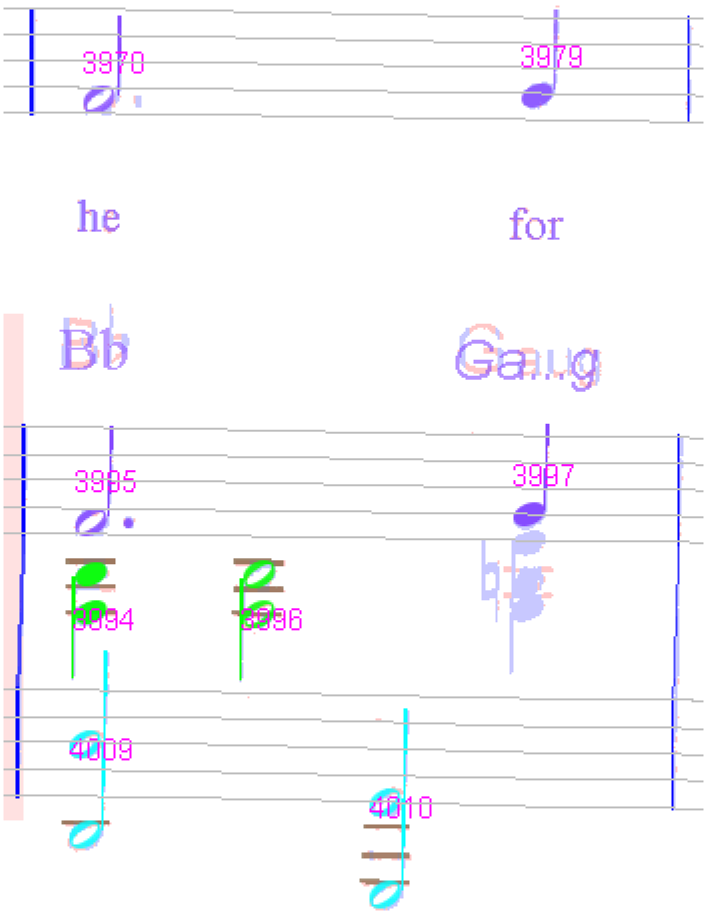
This gives the new picture below, where it is clear that we could merge chord 3963 and chord 3964 into a single chord:



Measure strip

A right-click within a measure N leads to a measure contextual menu which offers to print out all voices for the measure at hand.

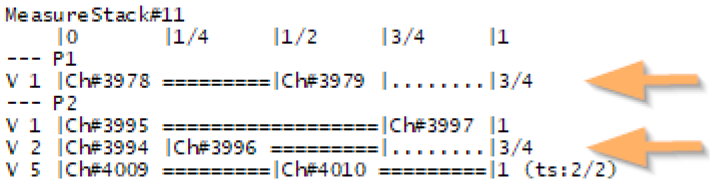
For example, let’s focus on the measure below:



`≡ Measure #N | Dump stack voices` prints a strip for the whole system-high measure stack.
`≡ Measure #N | Dump measure voices` prints a strip for just the part-high measure.

The latter (limited to one part) is meant for orchestra scores, with lots of parts within each system, to allow to focus on one part at a time.

Here is a measure strip at stack level (with part P1 and part P2):



```
MeasureStack#11
    |0        |1/4      |1/2      |3/4      |1
--- P1
V 1 |Ch#3978 ===========|Ch#3979  |.........|3/4
--- P2
V 1 |Ch#3995 =====================|Ch#3997  |1
V 2 |Ch#3994 |Ch#3996 ===========|.........|3/4
V 5 |Ch#4009 ===========|Ch#4010 ===========|1 (ts:2/2)
```

Strip legend:

- The strip top line displays the time offset of each slot within the measure.
- It then displays one part after the other, with one horizontal line per voice in

the part.

- A voice line gives the voice ID, then each chord ID vertically aligned with its starting slot.
- A “====” segment indicates the voice is active, while a “....” segment indicates the voice is inactive.
- A voice line may end with a string (ts:Num/Den) to indicate that this voice looks like a typical measure in a Numerator/Denominator time signature. [This could be used to infer the actual time signature, but this feature is now disabled].

When compared with the corresponding measure image above, the strip at hand shows 2 anomalies:

1. Voice P1/V1 ends at offset 3/4, while on the image we would expect this voice to end at offset 1.
   Moreover, Ch#3979 starts at offset 1/2, whereas on the image it starts at offset 3/4. (Evidently the dot on Ch#3970 has been overlooked by the engine.)
2. Voice P2/V2 also ends at offset 3/4. This is consistent with the image: there is a non-detected chord just below chord 3997.

Here again there is nothing, from the OMR engine point of view, to flag this measure as abnormal – all voices can be computed and all of them complete within the measure time limit.

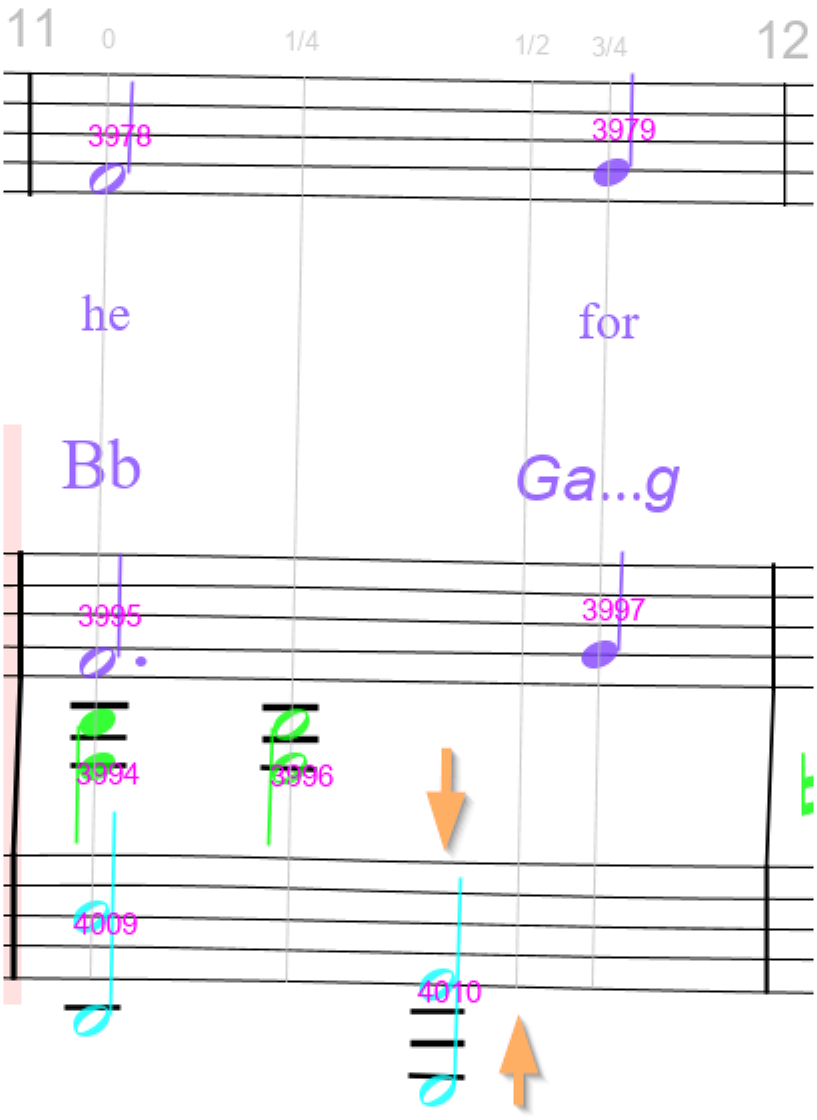
Time slots

Time slots are not visible in physical mode.

They are displayed in combined and logical modes – provided that the `View → Show annotations` flag is on.

A time slot is displayed as a thin vertical gray line, with its time offset value at the top.
Here below, we can see 4 slots with respective offsets:

Slot ID:	slot#1	slot#2	slot#3	slot#4
Time Offset:	0	1/4	1/2	3/4



In this picture, we can notice that slot #3 at offset 1/2 looks strange, with no chord aligned with it. Moreover, chord 4010 at bottom center is not aligned with any slot.

To be more precise, we can investigate the content of each time slot, via a right-click on the slot:

- `≡ Slot #3 | Dump chords` gives

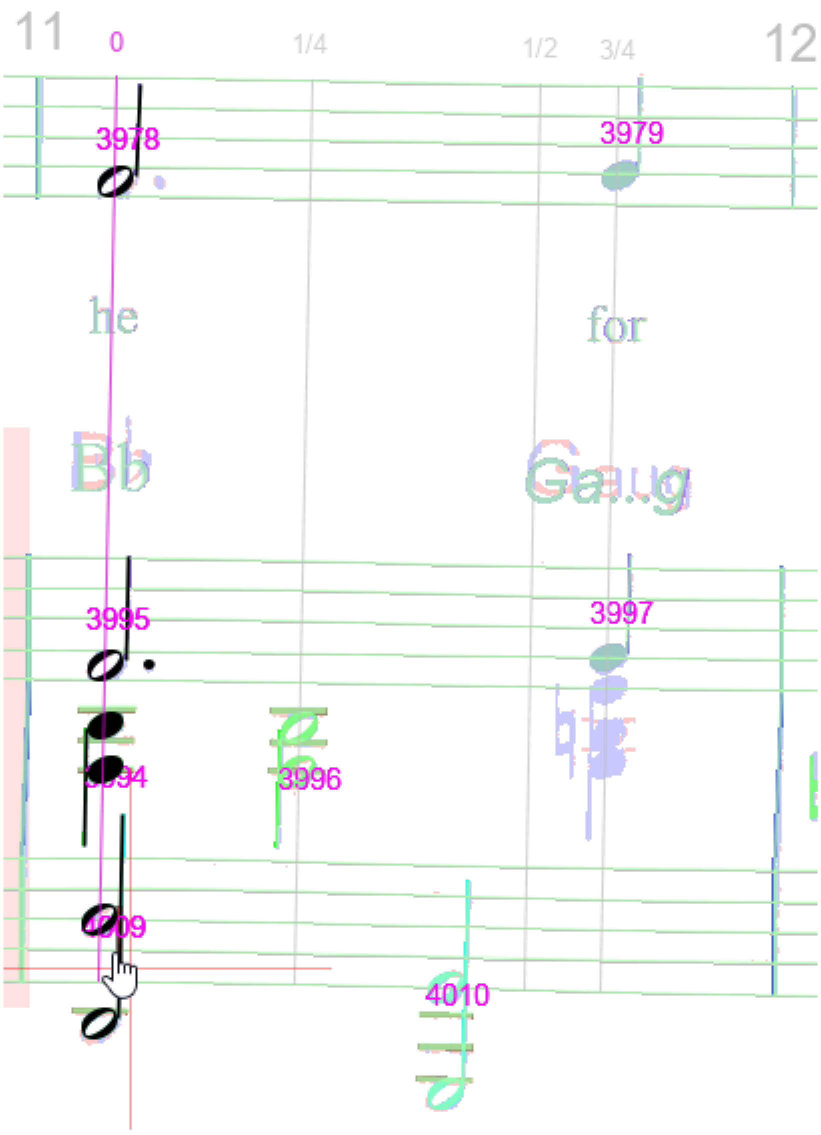
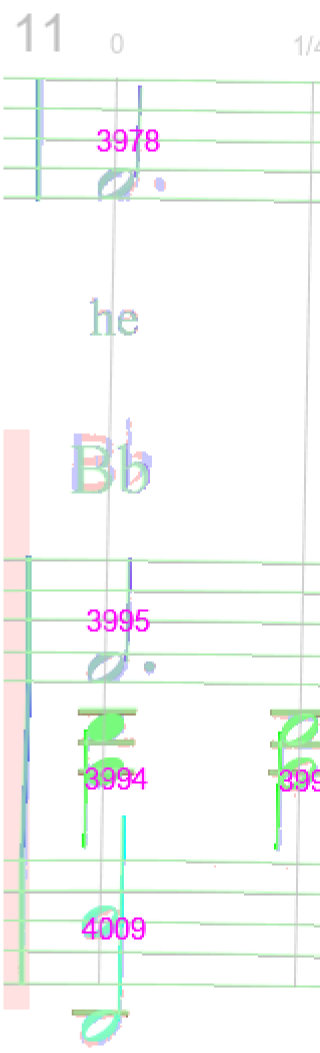
  ```
  slot#3 start=  1/2 [HeadChordInter{#3979(0.794/0.794) stf:7 slot#3 off:1/2 dur:1/4},HeadChordInt
  ```

- `≡ Slot #3 | Dump voices` gives

  ```
  slot#3 start=  1/2 [V1 Ch#3979 s:7 Dur=  1/4, V5 Ch#4010 s:9 Dur=  1/2]
  ```

These outputs are not very readable, but they say that this slot contains chords 3979 and 4010. Which looks strange.

A more readable output is available, but only in ♫ combined mode. We simply press the right mouse button and move it near a time slot, while staying vertically within part stave(s). This highlights both the designated slot and the slot-related chords as in the pictures below:

Correct slot #1 at offset 0		Abnorma
 11 0 1/4 1/2 3/4 12 3978 3979 he for Bb Ga.ug 3995 3997 3994 3996 4009 4010		

We can immediately see that slot at offset 1/2 wrongly contains chords 4010 and 3979.

A closer look near chord 3978 in slot #1 (time offset 0), at the upper left corner of the measure, indicates that there is an unrecognized augmentation dot on the right of the chord note head, which is the actual root cause of all the aforementioned time slot problems.

UI tools

TABLE OF CONTENTS

Interaction with the OMR engine

The OMR engine works as a sequence of 20 sheet steps, a step being a kind of mini-batch.
We, as an interacting user, can get in only at the end of a step.
[When a sheet first appears with its Data tab, the OMR engine has already performed the LOAD and BINARY steps.]

Following any user manual action, OMR data is immediately updated depending on the impacted steps, the sheet view is updated to reflect the new status of OMR elements directly or indirectly modified, and also to indicate any "*abnormal*" situations detected on-the-fly by the OMR engine.

- For example, a black note head with no compatible stem nearby, or vice versa a stem with no linked note head, will appear in red. Such a situation can happen temporarily while items are being manually created or edited, one after the other.
- Similarly, a whole measure may have its background colored in pink to signal a rhythm problem.

Tasks

Task sequence

Whenever we modify an entity in some way, this task is recorded as such, perhaps with some joint tasks, into a “*task sequence*” which is then performed on the targeted entities.

A task sequence can be pretty large. For example:

- We can select a dozen inters and ask for the removal of the whole set.
- We can remove an inter ensemble; this triggers the removal of all the members of the ensemble: words of a sentence, heads of a head-chord, etc.
- Removing the last remaining member of an ensemble also removes the now-empty ensemble.

A task sequence is *indivisible*, but we can **do** it, **undo** and **redo** it at will, with no limits. Only moving the engine forward (or pseudo backward) from one step to another clears the user task history.

Undo

The `Undo` command cancels the last task sequence (whether it was a do or a redo) with all its consequences.

- We press the **Undo** button ↩ on the tool bar,
- Or type `Ctrl+Z` (`Command+Z` for macOS)

Redo

The `Redo` command re-performs the last un-done task sequence.

- We press the **Redo** button ↪ on the tool bar,
- Or type `Ctrl+Shift+Z` (`Command+Shift+Z` for macOS).

When to interact?

Most user corrections can be done at the end of a sheet transcription (at the final `PAGE` step of the engine pipeline).

Some corrections are more effective at earlier moments; experience will tell us. Here are interesting stopping points, presented in chronological order:

-

SCALE step is an interesting stop if the beams are questionable.
That is, if we get a message saying that the beam thickness value is just "extrapolated", we must be careful! This tells us that beam-related data could not reach the quorum needed to infer a reliable thickness value.
So, let's measure the actual beam thickness on our own (using the Pixel Board) and modify the beam scale if needed (see the Sheet scale section).

- **GRID** step is where staff lines, then staves and systems are detected.
  Hence, this is the most convenient point to interact if we need to manually modify lines and staves (see the Staff editing section).

- **REDUCTION** step is where all candidate note heads are combined with candidate stems and beams and then reduced to come up with reliable notes – this does not include any flag or rest, which are addressed later in the SYMBOLS step.
  It is a key moment in the engine pipeline, because these "reliable" notes will never be called into question by the following steps. So much so that their underlying pixels will not be presented to the glyph classifier during the `SYMBOLS` step. So, if some of these notes are false positives, then it's much more efficient to correct them immediately, at the end of the REDUCTION step.

- **TEXTS** step is another key moment.
  It aims at retrieving all the textual items in the image and assigning them a role (such as title, part name, lyrics). The engine output for the `TEXT` step depends heavily on an external program (Tesseract OCR) which Audiveris has little control upon. It is hard for the OMR engine to decide if an item is really a piece of text, or if it should also be considered as a possible music symbol.
  Thus, it is efficient to manually remove text false positives at the end of this TEXT step to let their pixels be taken into account by the SYMBOLS step.

- **PAGE** step is where the OMR engine ends on a sheet and where we can observe and correct the final results.

TABLE OF CONTENTS

- Relation addition
- Relation removal
- Staff editing
- System merge
- Part merge
- Measure merge
- Key signature
- Time signature
- Chords
- Text
- Shared head
- Octave shift
- Tips and tricks

`Inter` addition

In the Audiveris data model, an `Inter` instance represents an interpretation, that is a candidate likely to become a musical symbol in the end.

A manually created `Inter` is flagged as `MANUAL` and cannot be called into question by the OMR engine, it is considered as certain.

We can manually create an `Inter` instance from a selected glyph, by “assigning” this glyph the desired shape. If we have no suitable glyph available, we can still create an `Inter` from nothing, via a simple drag & drop from the shape palette.

In both cases, we may have to specify the target staff for the created inter if the context is not clear enough for the engine.

TABLE OF CONTENTS

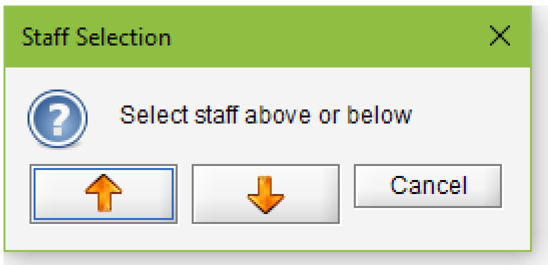
`Inter` from the underlying glyph

`Inter` creation from a glyph is the only case where a glyph is used in Audiveris manual editing.

The set of black pixels of one or several selected glyphs can be transcribed as one inter, by specifying the target inter shape. The precise location and bounds of the new inter are defined by the underlying set of pixels.

> **NOTE**
> This method of selecting a glyph to “assign” an `Inter` to it, requires of course that a **suitable glyph** be available. In we can’t find any such glyph, we’ll have to drag & drop the desired shape from the shape palette instead.

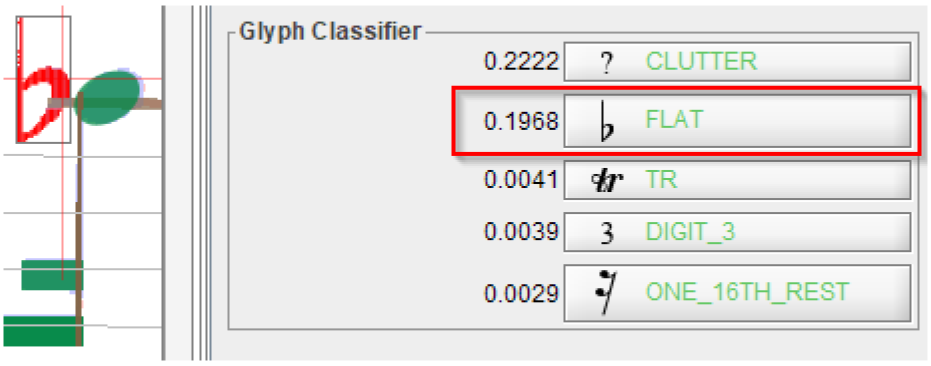
Also, knowing the target location is not always sufficient to detect the `Inter` **target staff**. Several heuristics are used by the software, but if they fail we will be prompted for the target staff:



Select staff above or below

The target shape can be specified via different means, as follows.

Via the glyph classifier



0.2222	CLUTTER
0.1968	FLAT
0.0041	TR
0.0039	DIGIT_3
0.0029	ONE_16TH_REST

This classifier automatically operates on the current glyph, be it the single selected glyph or a transient glyph built on-the-fly from a multi-glyph selection.

Chances are we’ll find our desired shape in this “top 5”. If so:

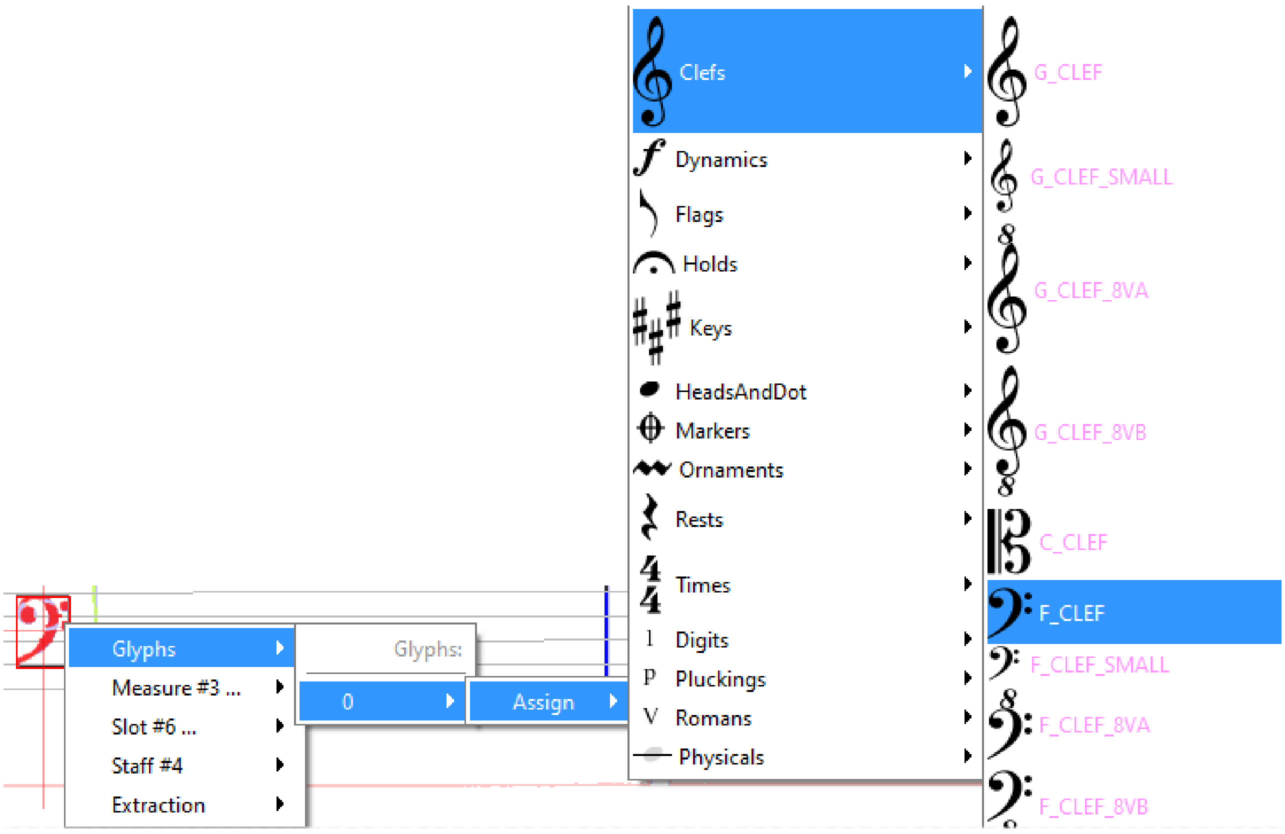
- We can just press the shape *button* with the mouse left button
- Or, on the keyboard, we can type the *number* (generally in 1..5 range) corresponding to the rank of our desired shape within the top 5.

NOTE
In order to use this classifier method, the target shape must be handled by the glyph classifier.
This is **NOT** the case for the following shapes:
- **Lines** (staff lines, ledgers, barlines, stems, slurs, endings) use ad-hoc methods,
- **Beams** use image morphological closing,
- **Note heads** use template matching,
- **Texts** use OCR.

NOTE
The top 5 is just a raw output of the glyph classifier, which runs on the glyph in isolation, that is ***without taking the neighborhood into account***.
This means that:
- a dot (augmentation, staccato, …) will be at best recognized as `DOT_set`,
- a half or whole rest as `HW_REST_set`, - a grace note or an 8th beat unit as `EIGHTH_set` . For these three cases – and only for them – the glyph classifier is not precise enough.
We’ll have to explicitly select a more specific target shape, via the glyph pop-up menu or via the shape palette.

Via the glyphs pop-up menu

Once a glyph has been selected, we use a mouse right click to access the `≡ Glyphs` sheet contextual menu.



Then we navigate through shape sets to our precise target shape.

Via the shape palette

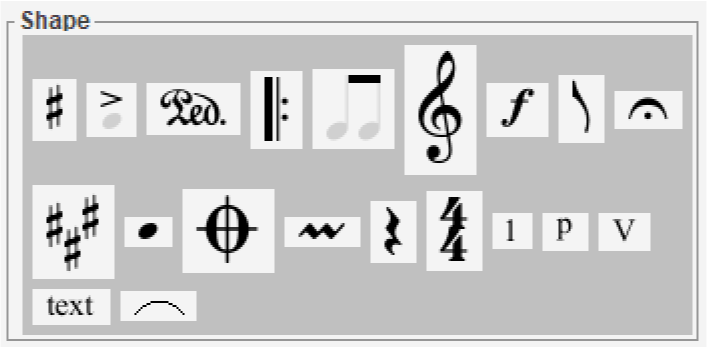
The shape palette (see next section) allows to assign a shape to the selected glyph. This is done by a double-click on the desired shape in the palette.

The shape palette

The palette offers the ability to choose the desired shape for a selected glyph.

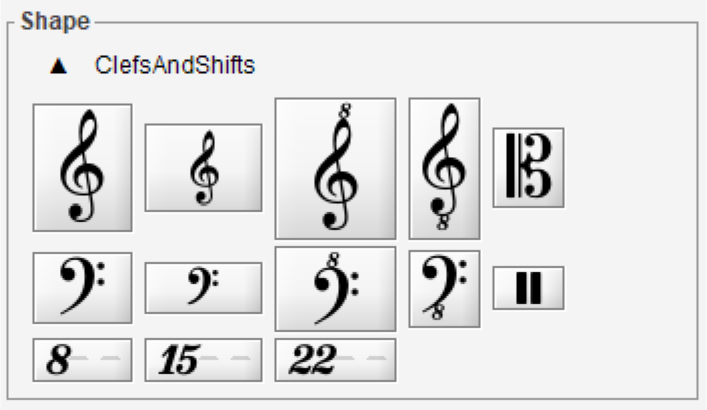
And even if no precise glyph can be selected, we can directly drag a “ghost” `Inter` from the shape palette, located on the right side of the sheet view, and drop this ghost at the desired target location.

The starting aspect of the shape palette is a catalog of all shape sets. It exhibits a dark-gray background, with one representative button for each shape set. Nothing can be dragged from this catalog, we must first select a shape set:



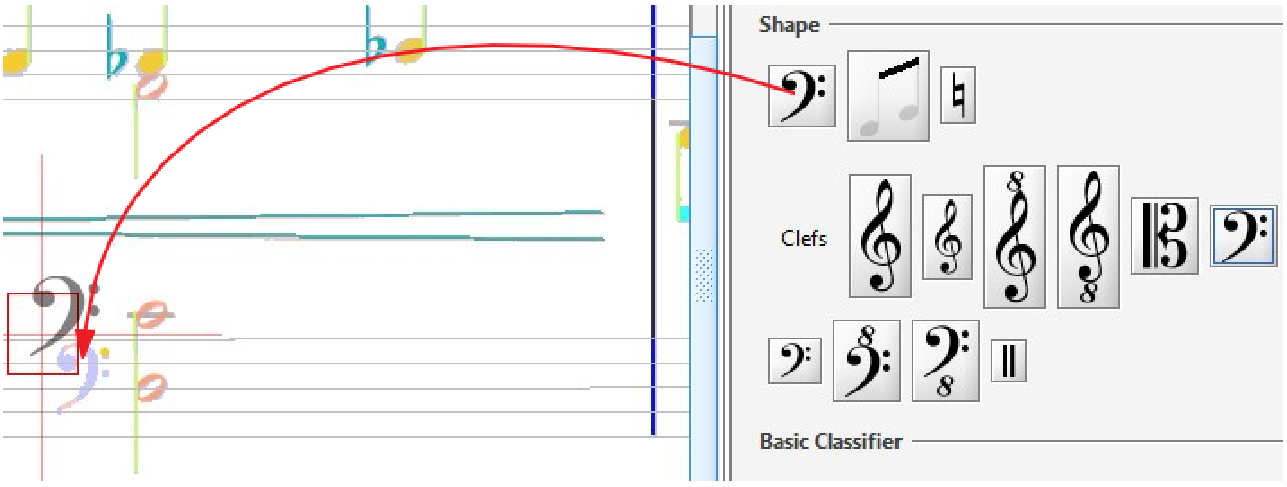
Entering a shape set

Pressing a shape set button replaces the catalog view by a specific palette dedicated to the selected shape set. For example, pressing on the `ClefsAndShifts` set button gives:



Within a set, a shape can be:

- Assigned (by a left double-click) if a glyph has been selected,
- Or dragged and dropped to a target location.

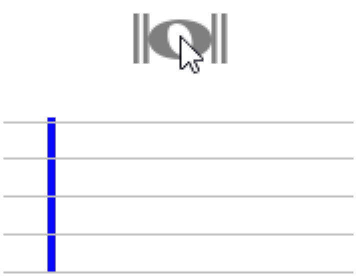
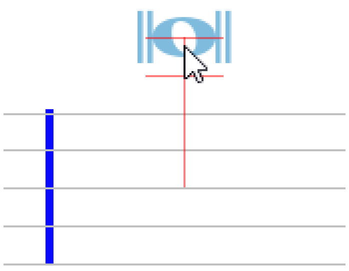


Selecting the target staff

While we are dragging a shape, we have the freedom to move it wherever we want. The last staff we hovered over is selected as the current target staff.

Before we “select” a staff, the dragged shape “ghost” is displayed isolated in dark-gray.

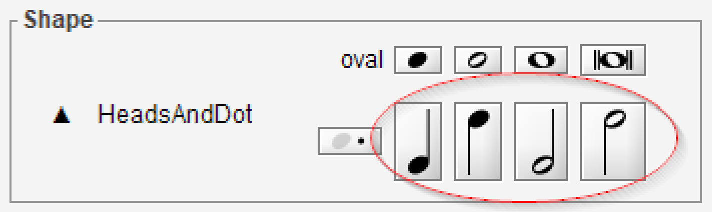
Once a staff has been “selected”, a thin vertical red segment goes from the shape center to the target staff middle line, the shape turns into the `Inter` selected color, additional objects can appear – such as intermediate ledgers or potential relations with nearby inters –, it may get snapped according to staff lines, etc.

before staff selection	after staff selection
	

We can drop the shape only when a staff target has been selected. If not, the drag & drop action is abandoned.

Case of compound shapes

Note the `HeadsAndDot` set now contains four new shapes located at the end.



These are quarter notes and half notes, with stem either up or down. There are called “compound” because they combine two shapes: a head shape and a stem shape.

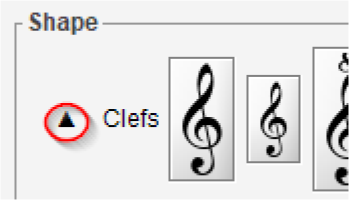
They are generally more convenient to insert as a whole, the head can still be snapped on staff line or ledger and the stem can still be automatically linked to beams nearby.

> **NOTE**
> Once dropped, such a compound shape is replaced by two separate Inters (head `Inter` and stem `Inter`) linked by a HeadStemRelation. We can then later edit each “part” separately, for example to modify the stem length. And we can add flags to the stem.

Exiting a shape set

To leave the specific set palette and go back to the catalog view, we can:

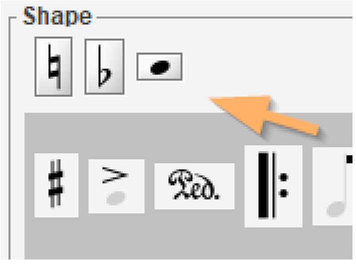
- Click on the “triangle-up” sign located on the left side of the set palette,



- Or press the `ESC` key on the keyboard.

Shape cache

We can notice, appearing on a line above the shape palette, the list of the most recent shape buttons we have used so far.



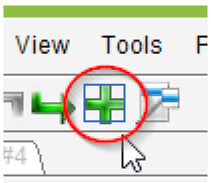
These *cached* buttons are meant for further direct access, avoiding the navigation to their containing sets.

And if we want to add many Inters of the same shape, we can consider using the “Repetitive Input” mode below.

Repetitive input

If we have numerous Inters of the **same shape** to add in a row, a convenient way is to switch temporarily to the “*Repetitive Input*” mode.

We click on the toolbar icon or select the menu item `Sheet → Toggle repetitive input` or use the shortcut `Ctrl+Shift+N` and this mode is now set on.



From that point on, pressing with the left-button anywhere on the sheet will add a new `Inter` (with the latest shape used) at the designated location. We can shift the `Inter` location precisely, then release the mouse when we are satisfied.

If we press the left-button again, yet another instance will be created, and so on. That's the rule, so let's mind our mouse press actions!

The latest `Inter` inserted is left in "`Inter` editing mode" with its editing handle(s) displayed. We can press the `ENTER` key to finish this edit, or click somewhere else to create another `Inter`.

This repetitive mode is meant for simple shapes – like a head, a rest, an accidental, etc. – that don't require any further resizing.

But what if we really need to resize the inserted `Inter`? We simply set the repetitive mode off, so that we can press and drag the inter editing handles. Then, we can set the repetitive mode on again if so desired.

To exit this rather specific mode, we toggle the mode (via the toolbar icon, the menu item or the shortcut).

Relations with other `Inter`'s

Key relation(s) with the nearby `Inter`(s), if any, will be updated automatically as we create – or later edit – the `Inter`, but only as long as the required geometrical relationships can apply (for example, as long as an accidental is sufficiently close to a note head on its right side).

If the relation constraints are not met, we will have to set the relation manually afterwards.

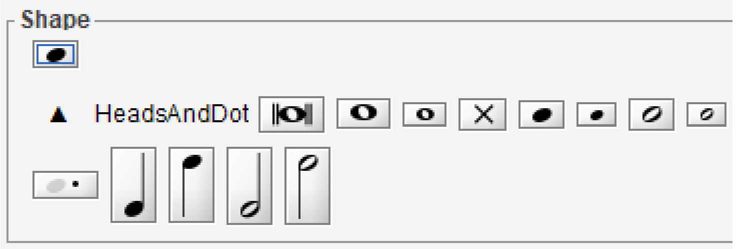
Shortcuts for inter addition

In order to make long editing sessions easier, there are a few shortcuts to assign interpretations without having our hands leave the keyboard.

They all work with a sequence of 2 strokes:

1. The first stroke selects a *set* of shapes.
   The selected set content appears in the right column.
2. The second stroke selects a *shape* within the current set.
   The selected shape appears first in the shape cache.

Example: Let's press `h` (heads) then `b` (black) and we get the `HeadsAndDot` set content displayed and the black head shape in the cache.



If we had a glyph selected beforehand, this glyph is assigned the selected shape. If not, no glyph gets assigned, but the shape cache now presents our selected shape in first position, ready for further use (via double-click, drag & drop or repetitive input).

Shortcuts table

Only sets and shapes that are used rather often are supported.

1st key	Set	2nd key
a	accidentals	**f** (flat), **n** (natural), **s** (sharp)
b	beams	**f** (full), **h** (half), **3** (triplet)
d	dynamics	**p** (piano), **m** (mezzoforte), **f** (forte)
f	flags	**u** (up), **d** (down)
h	heads	**w** (whole), **v** (void), **b** (black), **d** (augmentation dot), **h** (half-note), **q** (quarter-note)
p	physicals	**a** (slur above), **b** (slur below), **s** (stem)
r	rests	**1**, **2**, **4**, **8** (full, half, quarter, eighth)
t	texts	**l** (lyrics), **t** (text), **m** (metronome)

`Inter` editing

Since 5.2 release, any `Inter` – whether it has been created by the OMR engine, manually created from a glyph or dragged & dropped from the shape palette – can be edited in terms of precise location and size.

This is now the default general behavior, knowing that, depending on the `Inter` class at hand, our mileage may vary.

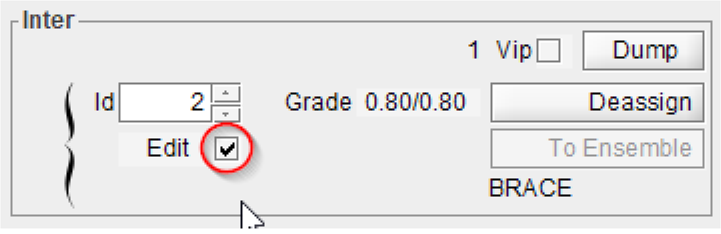
And an `Inter` being edited has a behavior identical to one being dragged & dropped, in terms of dynamic relations evaluation and in terms of potential snapping to items nearby.

TABLE OF CONTENTS

Entering editing mode

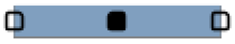
To start editing an `Inter`, we must first set it into `editing mode`:

- This is most conveniently done by a left double-click on the `Inter`.
- If the `Inter` is already selected, the InterBoard on the right displays information about it. In this board, the `Edit` checkbox can be used to set the `Inter` into editing mode.


Inter
1 Vip □
Id 2
Grade 0.80/0.80
Edit ☑
BRACE

`Inter` handles

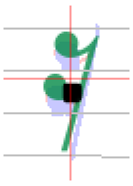
The `Inter` being edited now displays a set of one or several handles, depending on the `Inter` class.

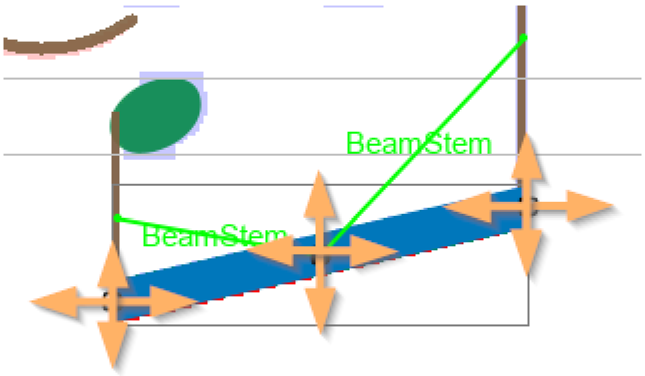
 Handles are displayed as small squares with rounded corners, filled in black for the current handle, in white for the others. Clicking on another handle makes it the new current handle.

A handle is meant to be moved:

- either by mouse dragging,
- or by pressing the keyboard `ALT+ ←/↑/↓/→` keys in the desired direction.

Clicking on any location other than `Inter` handles makes the `Inter` exit its editing mode and commits the editing.

 By default, just one handle is displayed as in the case of the 16th rest shown above, allowing the inter to be shifted in any direction. Since this inter exhibits only one handle, it offers no resizing capability – not surprisingly, we cannot resize a 16th rest symbol.



In the case of a beam, 3 handles are displayed. The center handle shifts the whole beam in any direction. The left or the right handle moves just this beam edge in any direction, leaving the other edge in place.

We will quickly notice, while moving or resizing this beam, that the green relation segments with the related stems can come and go according to the possible geometric connections between the beam being moved or resized and the (static) stems.

These are just two typical examples. See the `Inter` Editors reference section for a description of all available `Inter` editors.

Exiting editing mode

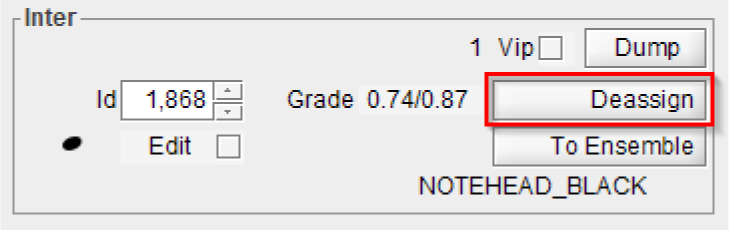
Finally, to complete the on-going editing, we simply press the `Enter` key or click on any location other than the `Inter` handles.

It is always possible to undo (and redo) any manual editing.

`Inter` removal

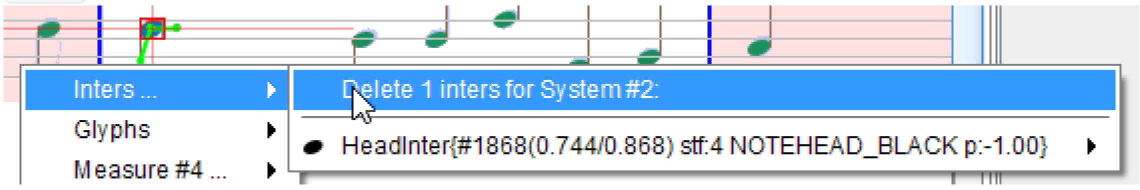
When one or several `Inter` instances have been selected, we can remove them as follows:

- The `Inter`-board has a `Deassign` button which removes the selected inter displayed in the board. This applies only for the displayed `Inter`.



Id 1,868
Grade 0.74/0.87
NOTEHEAD_BLACK

- The `≡ Inters` contextual menu provides an item to remove the selected `Inter` entities according to their containing system.


Inters ...
Glyphs
Measure #4 ...
Delete 1 inters for System #2:
HeadInter{#1868(0.744/0.868) stf:4 NOTEHEAD_BLACK p:-1.00}

- Pressing `DELETE` key or `BACKSPACE` key on the keyboard removes all the selected inters.

If more than one `Inter` is selected, we will be prompted for confirmation beforehand.

Removing a selected `Inter` (or a set of selected Inters) automatically removes the relations these Inters were involved in.

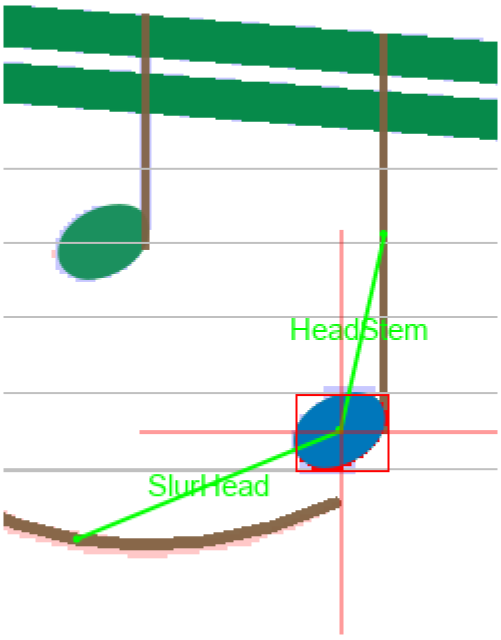
Removing the last member of an ensemble also removes that ensemble.

Relation addition

TABLE OF CONTENTS

Most `Inter` instances have relations with other `Inter` instances. For example the note head below exhibits 2 relations:

- a *SlurHead* relation between slur and note head,
- a *HeadStem* relation between note head and stem.



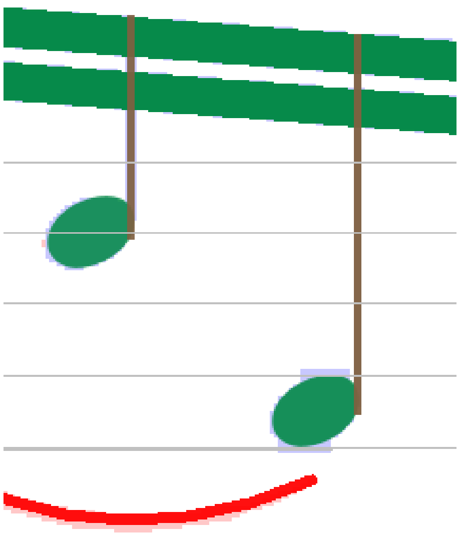
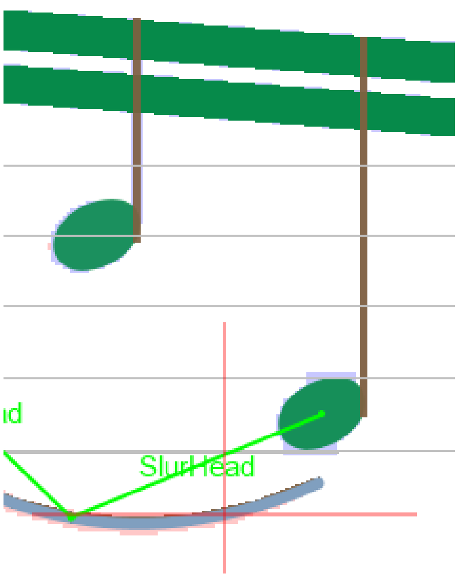
We can visually check the relations when we select an `Inter`. The name of each relation is also displayed, provided that the current zoom is high enough (>=2).

Mandatory vs. non-mandatory relations

Depending on the `Inter` class, an `Inter` instance may need a relation with another `Inter` instance.

If an `Inter` instance lacks a mandatory relation, it should somehow be removed before the end of the transcription process. This is the case when created as a candidate by the OMR engine but, if the `Inter` at hand was manually added, it can’t be automatically removed by the engine. It is simply flagged as “*abnormal*” and shown in red to call the user’s attention on it.

In the example below, a *SlurHead* relation between slur and note head is mandatory for the slur.

non-linked slur	linked slur
	

A missing relation can happen when the geometry rules are not matched, perhaps because the gap is a bit too wide between the slur and any note head. In that case, we have to either shift or resize the Inters accordingly or manually set the needed relation.

Note, regarding relation automatic search:

- A **mandatory relation** is automatically searched for, only at the moment the dependent inter (such as the slur in the example) is created, shifted or resized. This is so, simply because – using the same slur example – the slur *needs* a head but the head *does not* need a slur. A good practice, when several Inters have to be created manually, is to create the independent Inters first and the dependent Inters second.

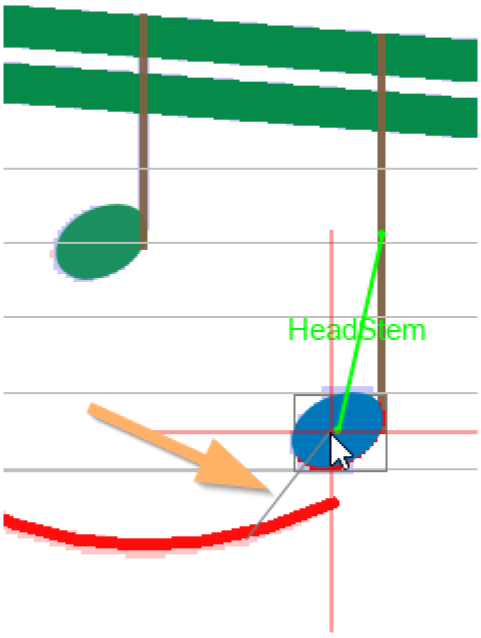
But still, when the geometry is really beyond some specified limits, the needed relation may not be automatically created and we'll have to add it manually.

- A **non-mandatory relation** is never searched for automatically. An example of a non-mandatory relation is the case of a direction sentence which doesn't always have a chord precisely above or below.
  For such non-mandatory relations, we have to decide if we set them or not.

Linking Inters

Assuming the slur is not linked to the note head, we need to insert the relation between them.

To do so, we can point and drag from the slur to the note head (a thin black vector will appear as we move the mouse, see picture below)



Then, we release the mouse when reaching the targeted head.

Audiveris will search among all the inters grabbed at the first and last locations, find the missing relation if any between those two collections of inters and set the proper relation.
This commits the relation insertion.

NOTE
When linking two elements, say A and B, the direction is irrelevant: we can either drag from A to B or from B to A, the result will be identical.

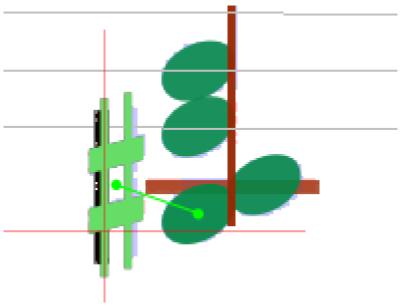
Relation removal

TABLE OF CONTENTS

Removing a wrong relation

There is no direct way to select Relation instances. They can be selected only indirectly, via the selection of one of the `Inter` instances they link.

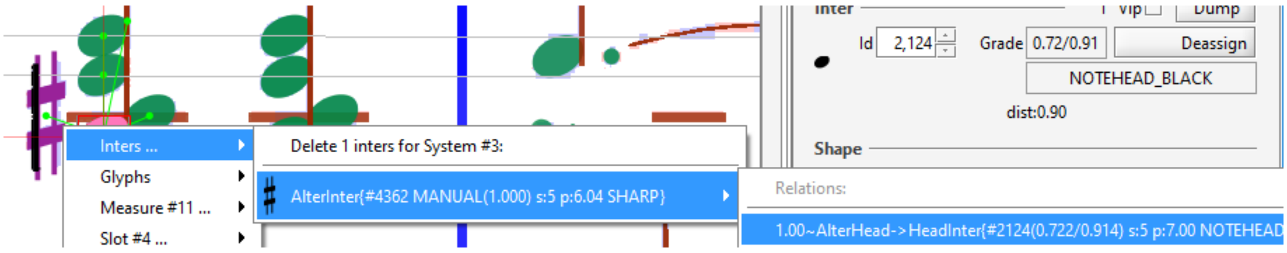
In the following example, a sharp sign has been linked to the wrong note head:



To select this relation, we can first select the involved sharp sign. This will result in the picture above.

Then we use a right-click to display the contextual menu `≡ Inters`, hover on the `Inters...` submenu, then on the sharp item to see the `Relations:` list of relations this `Inter` is involved in.

By clicking on the *AlterHead* relation, we will be prompted to confirm the removal of this relation.


Inters ...
Glyphs
Measure #11 ...
Slot #4 ...
Delete 1 inters for System #3:
AlterInter{#4362 MANUAL(1.000) s:5 p:6.04 SHARP}
Inter
Vip □
Dump
Deassign
Shape

Id 2,124

Grade 0.72/0.91

NOTEHEAD_BLACK

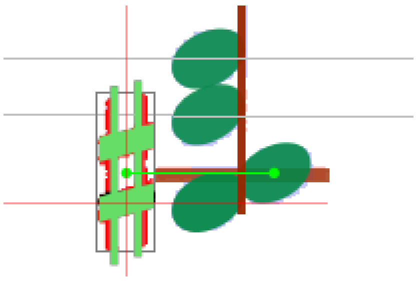
dist:0.90

Relations:

1.00~AlterHead->HeadInter{#2124(0.722/0.914) s:5 p:7.00 NOTEHEAD

Without this relation, the sharp sign is now no longer linked to any head, it thus appears in red abnormal status.

Finally, the correct relation should be manually added (see the Add Relation previous section) to result in the configuration below:



Implicit relation removal

In the precise case above (correcting reference of accidentals), explicit removal of the relation was not necessary, provided the correct relation is inserted manually.

This is so, because an accidental can reference only one note head (if we except the special case of a note head shared by two voices).

So the wrong *AlterHead* relation would be removed automatically when inserting a new one.

The same applies to note heads: they can reference only one stem (still excepting the special case of a single note head *shared* between two opposite stems). Here again, inserting a new *HeadStem* relation would remove the former one.

Staff editing

During the `GRID` step, the OMR engine strives to detect sequences of equally spaced, long and thin horizontal filaments. These filaments are then converted to staff lines and their underlying pixels are erased.

For this to work correctly, the score image must exhibit enough “long and thin” filaments.
Of course, there are musical symbols lying on these staves, typically beam symbols, which may hide the underlying staff lines. Generally, the engine can interpolate – or even extrapolate – the holes found in image regions with such broken filaments.

If the engine result is not satisfactory, we can manually correct these staves via staff editing, which is provided in two different modes: `Global` mode and `Lines` mode. The former operates on the staff as a whole, the latter on any line separately.

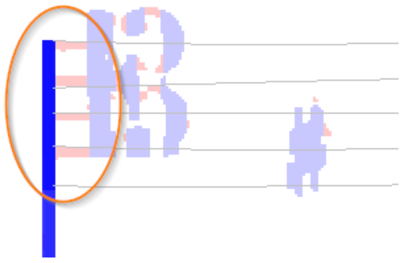
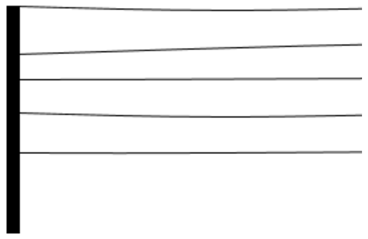
To enter staff editing, we right-click within a staff area, and in the `≡ Staff N°` contextual menu we select:

- either the `Edit staff` item to edit the staff globally,
- or the `Edit lines` item to edit each staff line individually.

TABLE OF CONTENTS

Lines mode

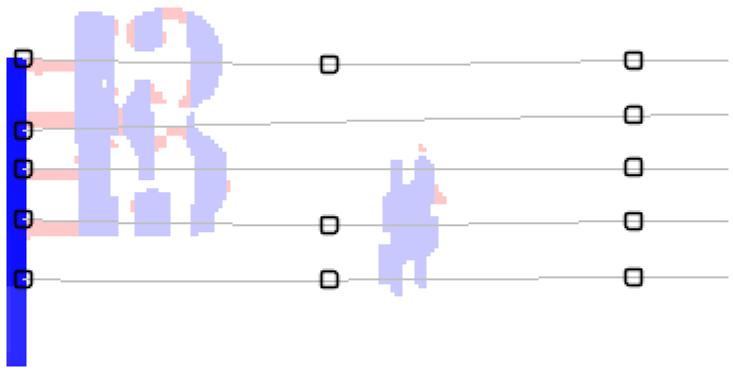
In this mode, we can modify the different lines of the staff individually.

Input view	Output view
	

In the example above, on the left-side picture, we can see that pixels at the beginning of the staff have been left over (these are the horizontal sections displayed in pink color). These non-erased pixels may impede further detection and recognition of symbols, because they will be considered as parts of the symbol to recognize, here a C clef.

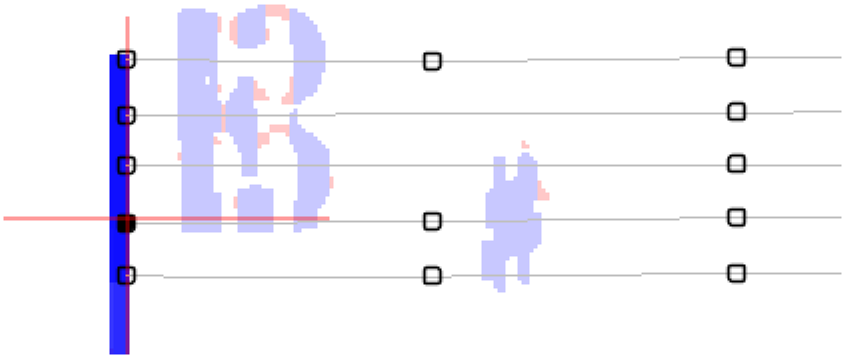
This was certainly due to the wrong extrapolation of staff lines, which is more visible on the right-side picture. The staff lines, as detected, are not evenly spaced near the barline.

To fix this, preferably right after the `GRID` step, we enter the staff editing in `lines` mode (Right-click in staff > `Staff#n` > `Edit lines`).



Each line of the staff now exhibits its own sequence of handles, which we can press and drag to modify the staff line locally.

In this `Lines` mode, the handles can move only vertically. In the case at hand, we could slightly drag some left handles up or down. We notice that the pink sections disappear when they get crossed by a staff line.



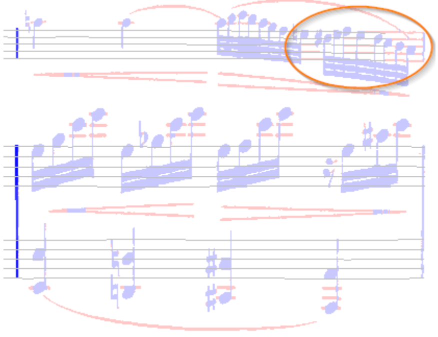
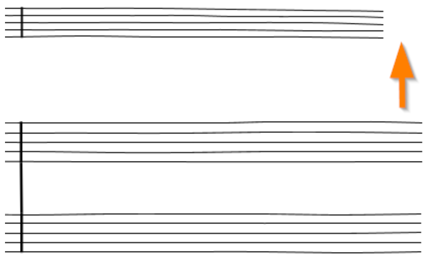
Clicking outside of any handle completes the editing.
We can always undo/redo the operation.

In short, this mode is meant to finely adjust the vertical location of any line portion. If we want to move the staff limits horizontally, then we need to use the `global` mode instead.

> **NOTE**
> The number of handles may vary from line to line, as can be seen in the picture. This number depends on the “wavyness” of the line detected by the engine. It has no negative impact on the editing capabilities.

Global mode

In this mode, we can move staff portions vertically and/or horizontally, and in doing so we modify all staff lines as a whole.

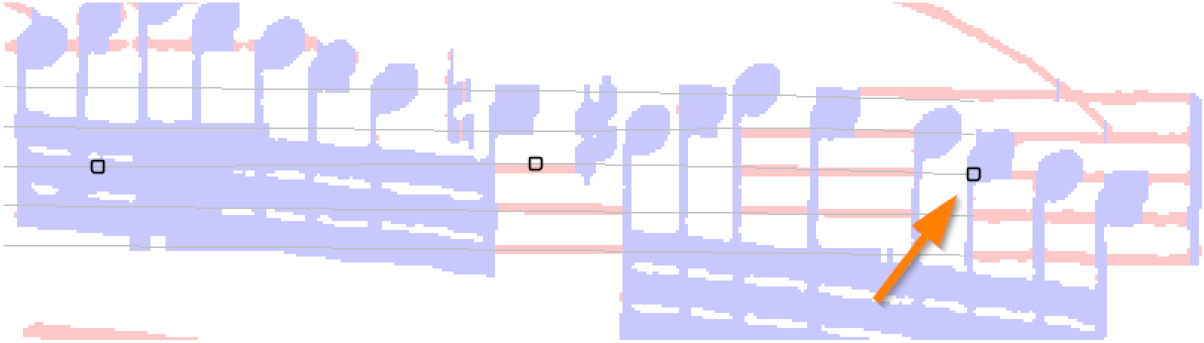
Input view	Output view
	

In the example above, the upper staff is so crowded with heads, stems and beams that the engine could not detect enough line portions.

So much so that many pink sections are still visible where staff lines should be, and that the staff has even been truncated on its right side.

To fix this, we enter the staff editing in `global` mode (Right-click in staff >

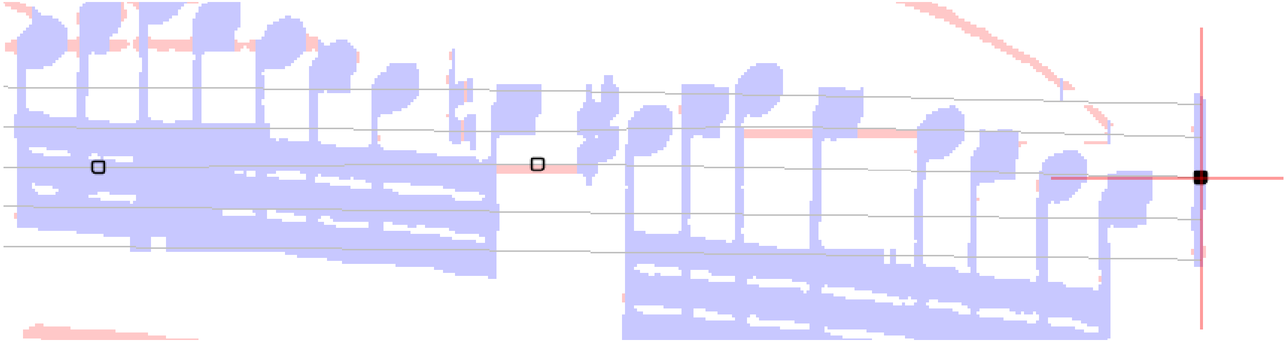
`Staff#n` > `Edit staff`).



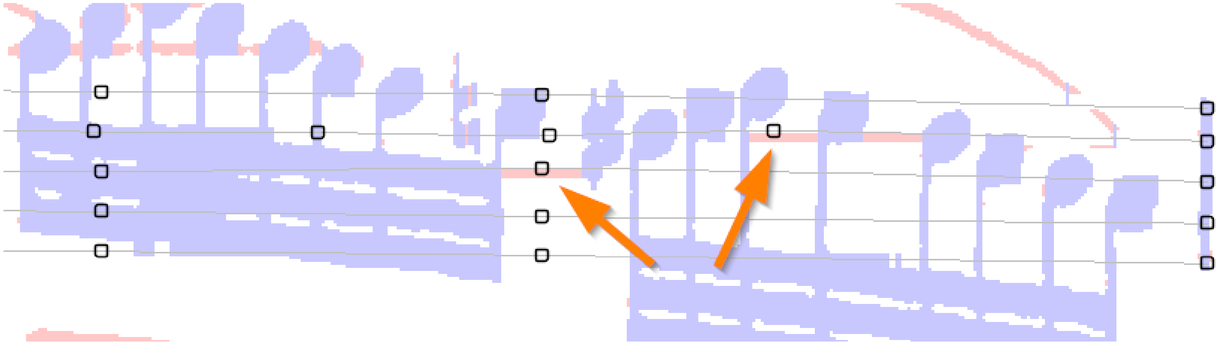
As opposed to the `lines` mode, we get handles only along the staff middle line. This is enough to move the whole staff:

- Side handles can move vertically **and horizontally**,
- Other handles can move only vertically.

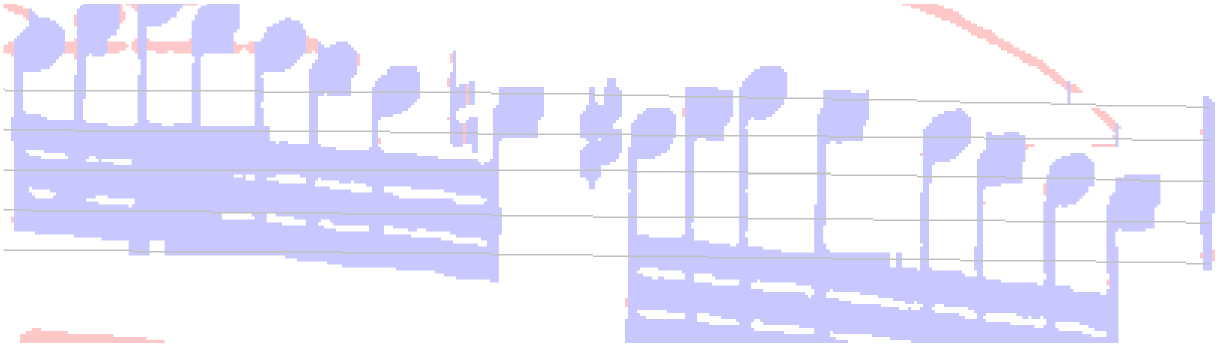
So, we press the right-most handle and drag it horizontally to the right until it reaches the right barline and vertically so that most if not all pink sections disappear.
We then release the mouse.



We still have a couple of pink sections to fix on lines 2 and 3 (counted top down). This is a job for the `lines` mode. We simply right-click in the staff area, and select the `lines` mode.



We slightly drag down two handles, and voila!:



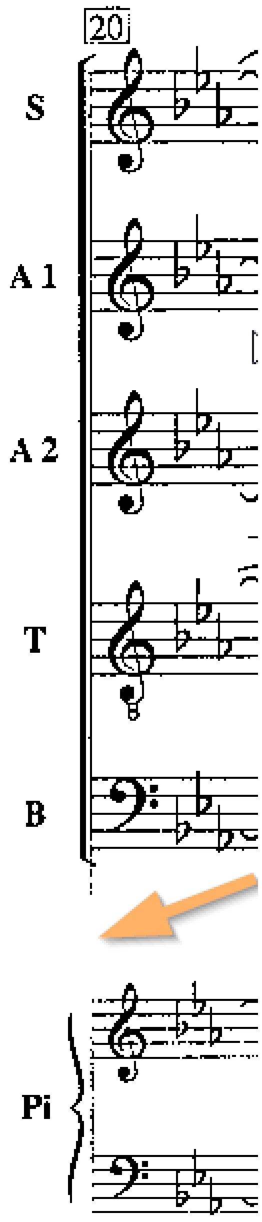
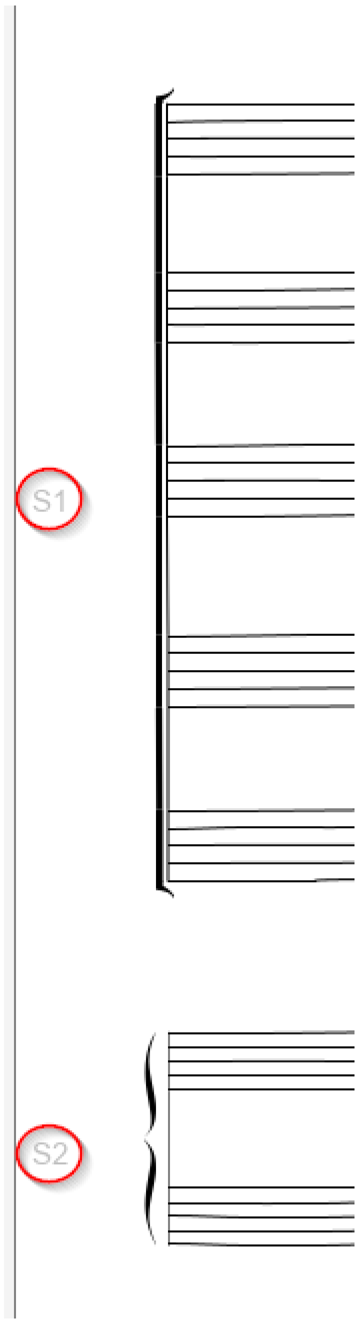
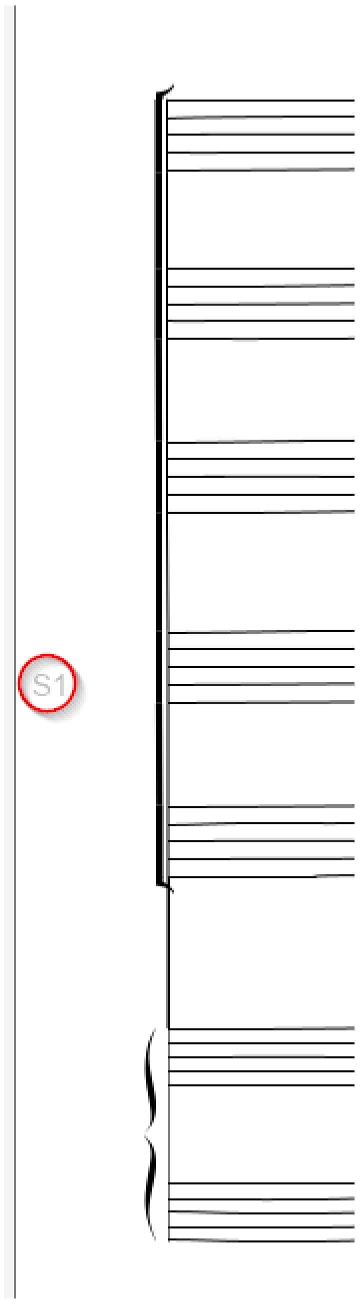
TIP: Since moving a handle is often a matter of very few pixels, we may find it more convenient to move a handle via the **keyboard**: While keeping the `Alt` key pressed, we can use the 4 arrow keys to move the selected handle one pixel at a time in the desired direction.

System merge

In the Audiveris `GRID` step, detected staves are gathered into systems, based on barlines found on the left side of the staves.

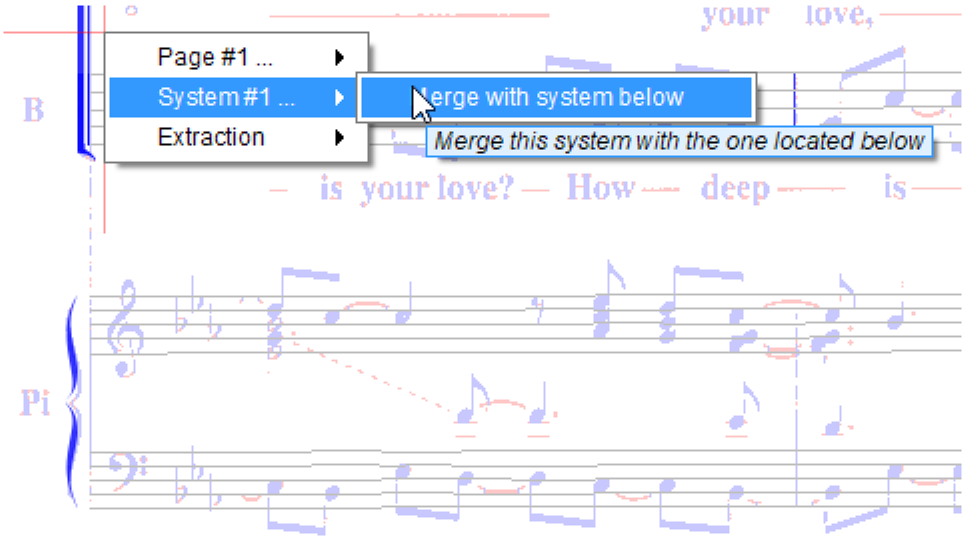
In a poor quality score image, many black pixels may have disappeared, sometimes leading to broken barlines.

In the example image below, the leading left barline has been damaged, resulting in a wrong detection of systems by the OMR engine.

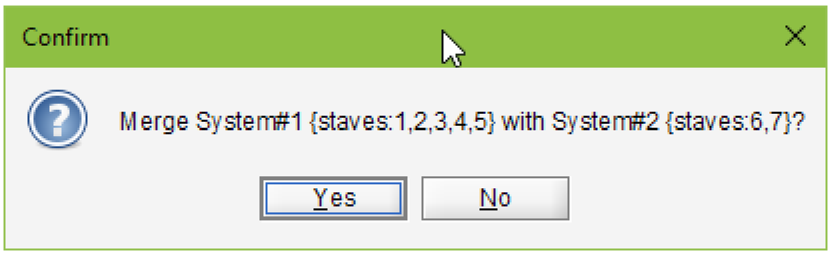
Left barline broken	Resulting grid before fix	Resulting grid after fix
		

Since the 5.2 release, we can manually fix this problem.

We point at the upper system portion, and via the right-click `≡ System N°` menu we select “*Merge with system below*”.



This is a key operation, so we need to confirm the detailed prompt:



Merge System#1 {staves:1,2,3,4,5} with System#2 {staves:6,7}?

And it’s done: a connector was created between the two barline portions and the two system portions merged.

We can still undo/redo the operation.

Part merge

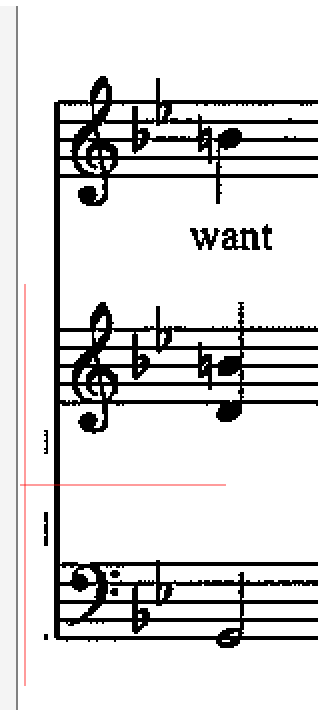
During the `GRID` step, the OMR engine detects staves and assigns exactly one part per staff, unless:

- There a multi-staff brace on the left margin of the staff, and
- The staves joined by the brace are also joined by connectors further along the staves (a connector is a vertical concrete segment joining two barlines).

If these two conditions are met, the two staves (more rarely three staves) are considered to refer to the same instrument (e.g. piano or organ) and thus to a single common part.

Some score images exhibit damaged braces, impeding the detection of a common part.

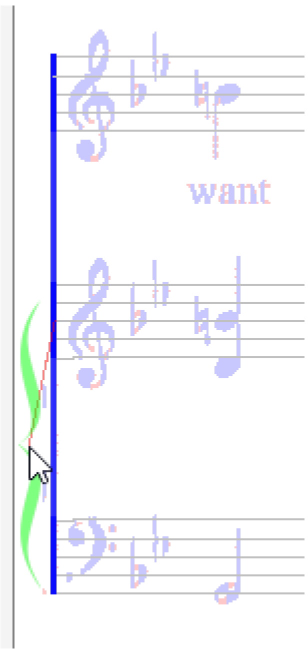
This is the case in the following example, where a poorly done scan has cropped an important portion on the image left side (the faint red cross is just the current location as given by the user):



So, this leads the OMR engine to detect a system with 3 parts.

To fix this, we manually drag & drop a brace `Inter` from the shape palette to where there should be a brace.

Let’s pay attention to hover over a staff of the target system. The brace ghost will turn from dark-gray to green, with a red segment going from brace center to staff middle line:



We can now drop the brace. The drop will commit the merge of the two embraced staves into a single part.

We can undo this operation (or remove the manual brace, which is equivalent).

> **NOTE**
> This is an *ad hoc* feature, meant to fix a brace cropped out. It works only by inserting a **manual** brace.
> We can still slightly shift or resize the brace if so desired. But let’s not try to extend the brace to a 3rd staff, this wouldn’t work.

Measure merge

A right-click in any measure stack allows us to merge this measure with the following one on the right, via the contextual menu `≡ Measure #N → Merge on right` (which is disabled for the last measure of a system).

It is a convenient way to remove all the intermediate barlines, if any, and merge the two measure stacks into a single one.

TODO: provide a concrete score example.

We can undo/redo this operation.

Key Signature

A key signature generally appears at the beginning of a staff, within what Audiveris calls the staff “header”. A staff header is an initial sequence of:

1. mandatory clef,
2. optional key signature,
3. optional time signature.

Generally, the engine correctly handles a key signature in the header, simply because it knows precisely where to look.

Later down the staff, generally right after a double barline, a key change may appear.

Before the 5.8 release, this key change was not handled by the engine. We had to always enter the key manually.

Since the 5.8 release, all key changes should be correctly handled by the engine. The editing features are still available for any manual correction, applied either to the initial key signature or to a key signature change.

Also since the 5.8 release, a key signature (initial or change) can be assigned or inserted as a whole or be built from its members.

TABLE OF CONTENTS

Data models

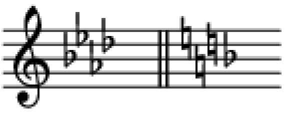
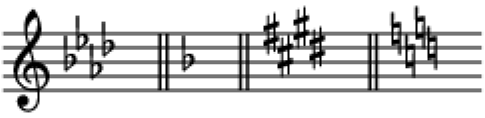
An **initial** key signature is an homogeneous sequence of 1 through 7 sharp or flat signs (located at precise pitches according to the effective clef and the key). There can’t be any natural sign in the initial key signature.

A key signature **change** is the sequence of:

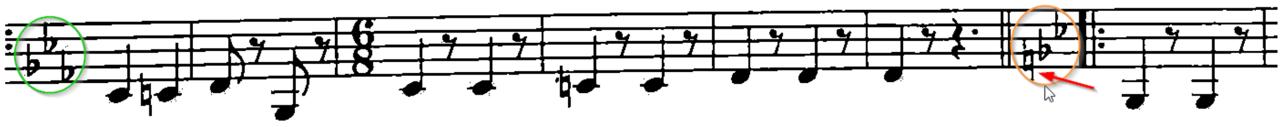
1. an optional sequence of 1 through 7 natural signs
2. an optional homogeneous sequence of 1 through 7 sharp or flat signs

In the following examples (stolen from Wikipedia), see the right-most signatures.

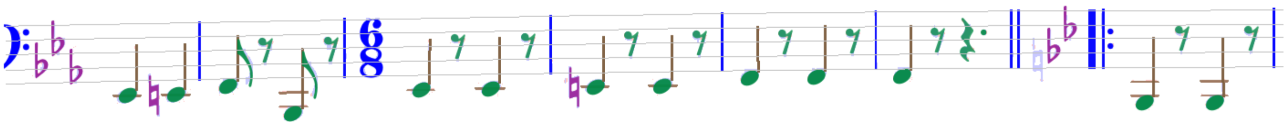
- In the first example, the natural signs are just courtesy signs and are safely skipped by the Audiveris engine, which recognizes just a 1-flat key signature
- In the second example, the natural signs are needed to cancel the previous 4-sharp key signature, and the Audiveris engine does process them as a cancel key

Example	**Description**	**Engine behavior**
	Courtesy natural signs	Naturals are ignored
	Mandatory natural signs	Naturals are recognized

Here is a concrete input:



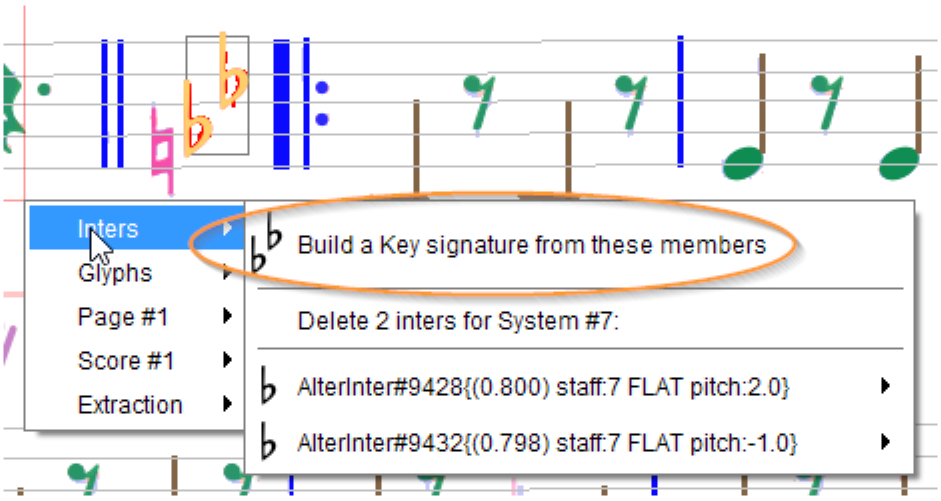
And the engine result:



Building a key from its members

Suppose we have a set of AlterInter instances (either recognized by the engine, or assigned manually).

Since the 5.8 release, we can select these inters and (if the sequence is compatible with the current clef / key configuration) a new item appears in the Inters popup menu proposing to create a key with these member inters.


Inters
Glyphs
Page #1
Score #1
Extraction
Build a Key signature from these members
Delete 2 inters for System #7:
AlterInter#9428{(0.800) staff:7 FLAT pitch:2.0}
AlterInter#9432{(0.798) staff:7 FLAT pitch:-1.0}

NOTA: Pay attention not to grab any leading courtesy natural sign(s) in the selected set, unless this is an all-natural cancel key.

Inserting a whole flat or sharp key

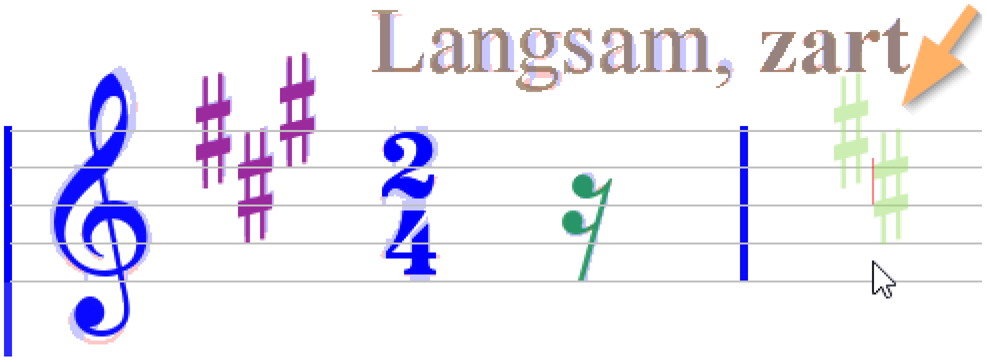
This can be done:

- Either by selecting a compound glyph and assigning the desired key shape,
- Or by dragging and dropping the desired key shape from the shape palette.

When dragging a “ghost” key from the shape palette, the ghost turns from dark-gray to green when we enter a staff and, as usual, a thin red segment is drawn from the ghost center to the staff mid line.

Moreover, the dragged key snaps immediately to the proper vertical position, according to the effective clef at the point of insertion.

For example, let’s insert a 2-sharp key, into the measure that follows the measure with a 3-sharp key:

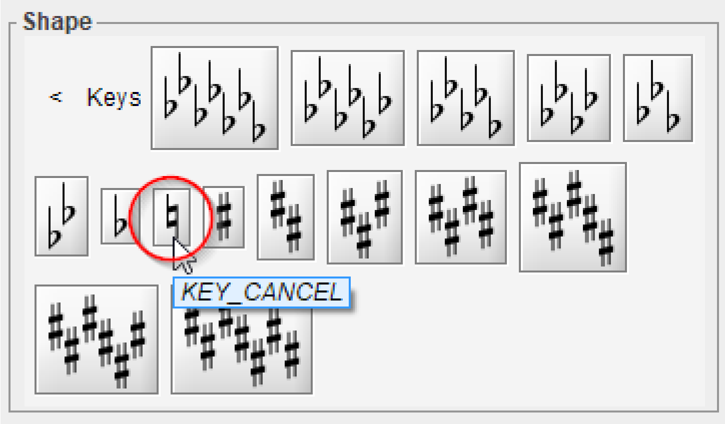


Note the two sharp signs are located on F and C steps respectively, and they can move only horizontally until we release the mouse.
Once dropped, we can still set the key into editing mode and again shift the inserted key.

Inserting a whole cancel key

We cannot insert a key made of courtesy naturals followed by sharp or flat signs. However, we can manually insert a *cancel key* – i.e. an all-natural key.

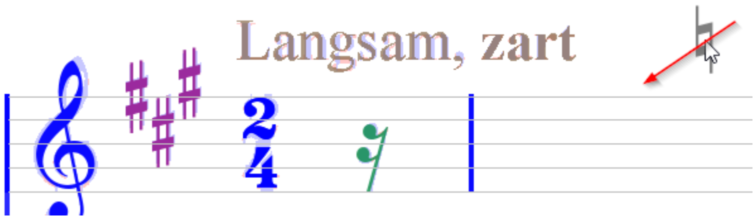
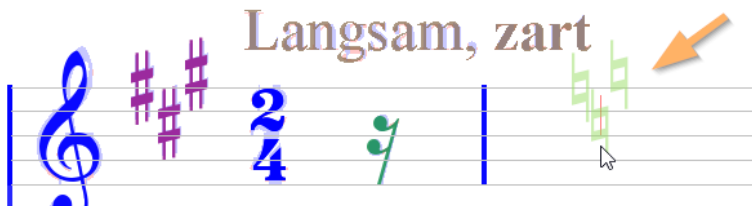
To do so, we drag the 1-natural key from the shape palette and move it to the desired insertion point.



When we enter the target staff, the 1-natural ghost key will turn as usual from dark-gray to green, but its configuration and position will be dynamically updated, to fit both:

- The effective **clef**,
- The effective **key** to be cancelled.

If for example the effective key is a 3-sharp key, the “cancel” key will be a 3-natural key, with each natural sign located according to the corresponding sharp sign to cancel:

Staff-relative location	Cancel Key appearance
Outside:	
Inside:	

We can notice that this works only when the ghost is located in a staff measure *different* from the staff measure that contains the key to be cancelled. This is so because there can’t exist two keys in the same staff measure.

Time signature

TABLE OF CONTENTS

Whole vs. pair

Audiveris OMR is able to handle both whole- and pair- time signatures:

- A *whole* time signature is a signature handled as a whole. It can be:
  1. A single musical symbol (COMMON_TIME and CUT_TIME)
  2. A predefined combo. As of today, these are 4/4, 2/2, 2/4, 3/4, 5/4, 6/4, 3/8, 6/8 and 12/8
  3. The *custom* combo (initially set to 0/0)
- A *pair* time signature is a signature handled as an ensemble of two separate time numbers. Not all pairs can be recognized, only the equivalent forms of predefined combos (see above).

The OMR engine can automatically recognize wholes and pairs, that is, all the examples above except the custom combo which is reserved for user manual definition.

For say a 2/4 time signature, the engine may recognize the “2” glyph and the “4” glyph leading to a pair, and at the same time recognize the “2/4” compound glyph leading to a whole combo.
The better grade will determine which interpretation will be kept. In fact, the precise chosen interpretation has no impact on OMR processing.

Predefined time signatures

If a time signature has not been detected or has been assigned a wrong value, we can select the underlying glyph and assign the desired value, either via a double-click in the shape palette or via the `Glyphs...` pop-up menu. This works for the whole signature (or for each signature part, in the case of a pair signature).

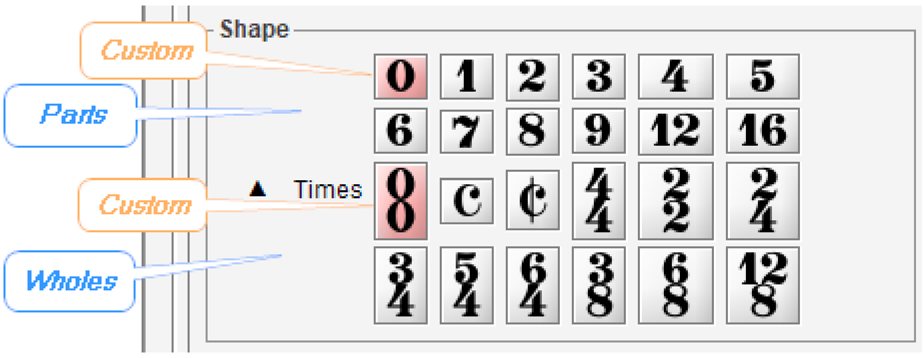
We can also drag a chosen time signature from the shape palette and drop it at the proper location, but this feature is available only for a *whole* signature; we cannot drag & drop time parts of a pair.

Moreover, in these two cases (glyph assignment and drag & drop), please note that the choice is *limited to the predefined shapes* of the palette.

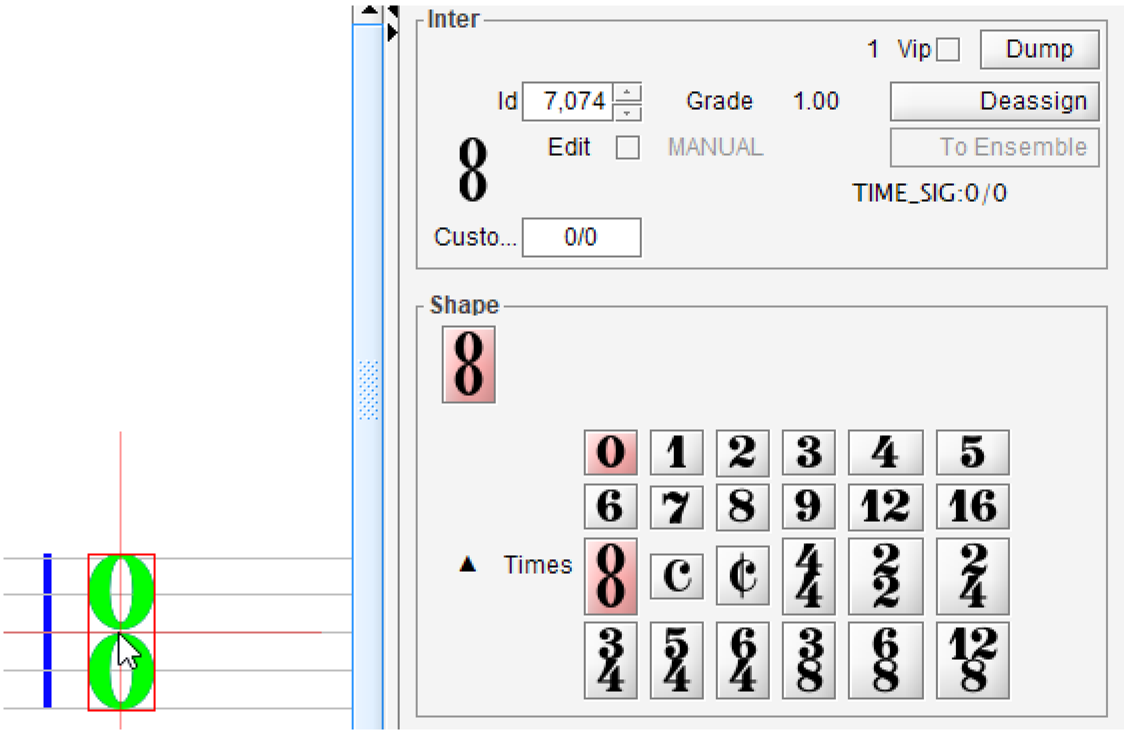
Custom time signature

Since the 5.2 release, the time shape palette provides a `custom` time signature. This is a fully-customizable combo signature.

To define a precise time signature, the most convenient way is often to drag the `custom` combo (0/0) from the shape palette to a target location:



Once dropped, the custom combo displays its initial value (`0/0`):



Inter

1 Vip □

Id 7,074 Grade 1.00

Edit □ MANUAL

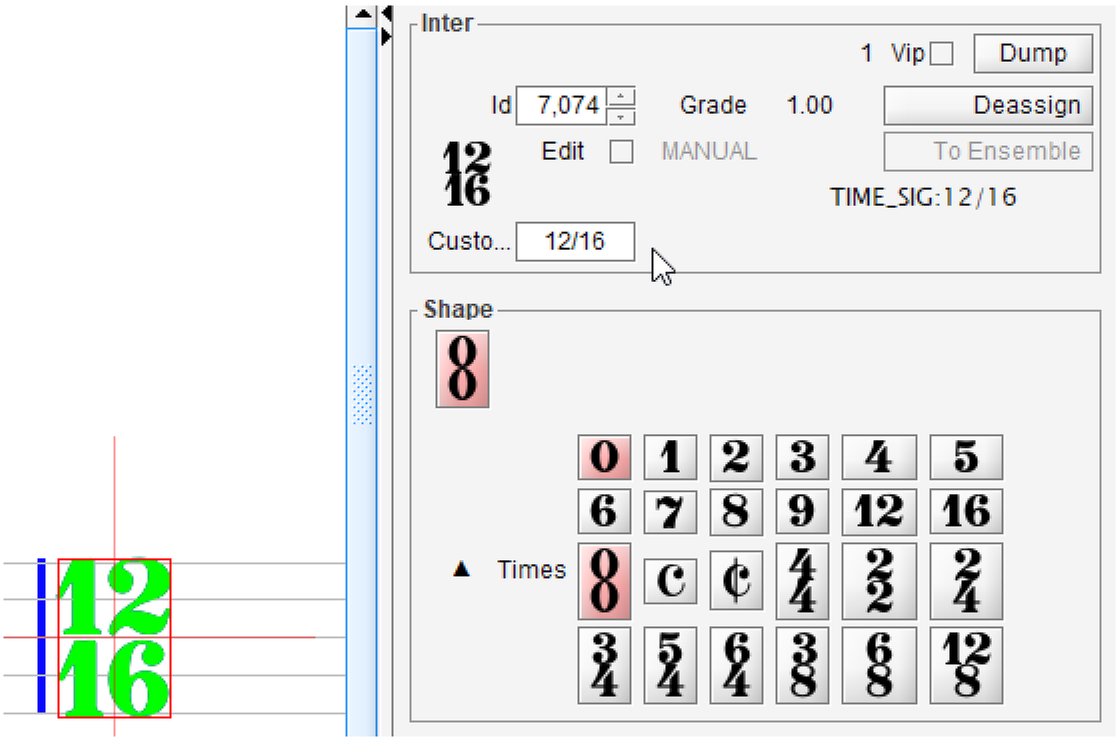
TIME_SIG:0/0

Custo... 0/0

We can then use the inter board to enter the desired values for our custom combo, formatted as “numerator / denominator”.

We are **no longer limited** to the predefined time values.

The only constraint is that each numerator or denominator value must stay within [0..99] range (0 being just a temporary value, of course).



Inter

1 Vip □

Id 7,074 Grade 1.00

Edit □ MANUAL

TIME_SIG:12/16

Custo... 12/16

Location

We cannot resize a time signature, and its location is snapped on the containing staff.

A whole time signature can be shifted horizontally within its staff.

In the case of a pair of parts, each part can be (slightly) shifted horizontally.

> Although in theory each staff of a containing system should have its own time signature, Audiveris can work with just one staff populated with a time signature.
> We thus don’t have to repeatedly enter the same time signature on each and every staff of a system.

Chords

For Audiveris, a *chord* is a container:

- A rest-chord contains exactly one rest.
- A head-chord contains one or several head(s), and often a stem.

Hence, we should understand by now that clicking on a note (head or rest) selects that note rather than its containing chord (this is the rule known as “*member over ensemble*”).

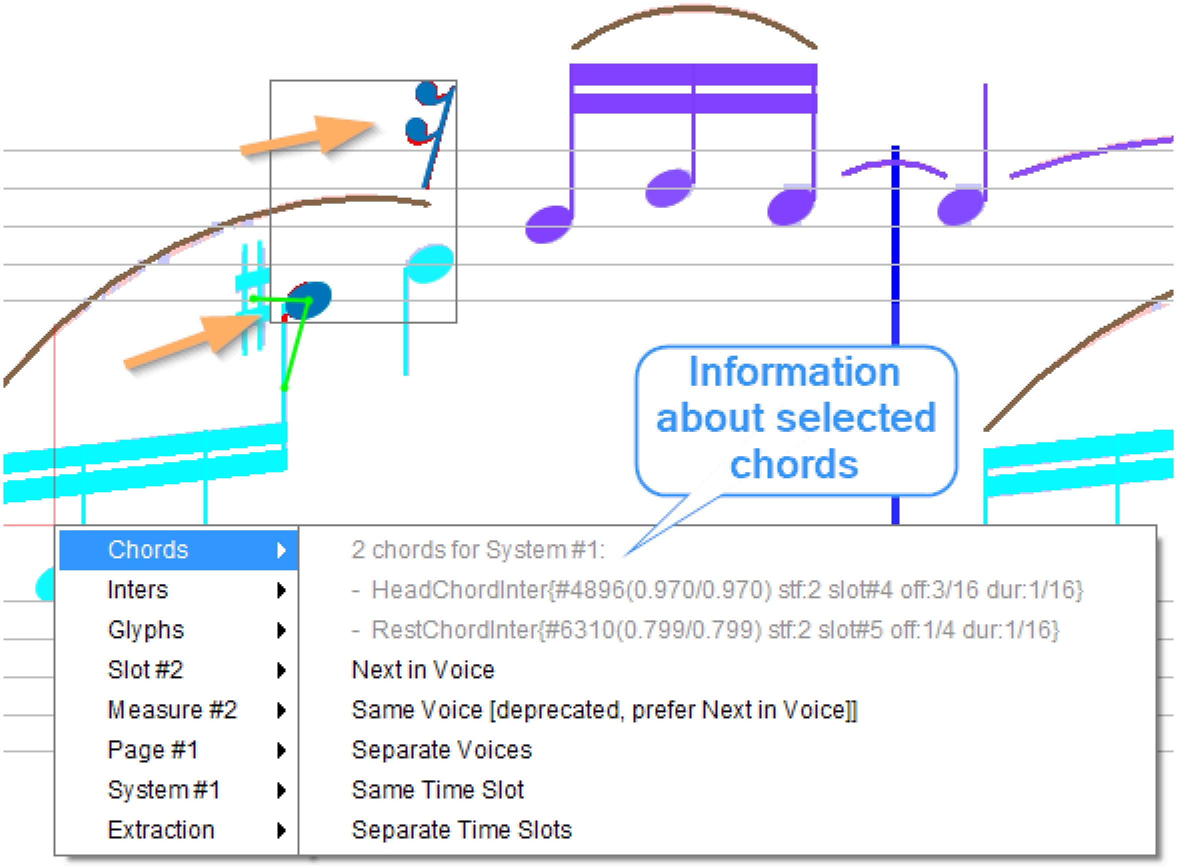
However, by selecting one or several notes, we can indirectly select and act on these chords.

This is made possible via the usual pop-up menu which can provide a specific `Chords...` sub-menu, whose content tightly depends on the chords configuration.

TABLE OF CONTENTS

Chords menu

Below, we have selected two notes as indicated by the arrows: one note head and one rest, before opening the `≡ Chords` contextual menu:

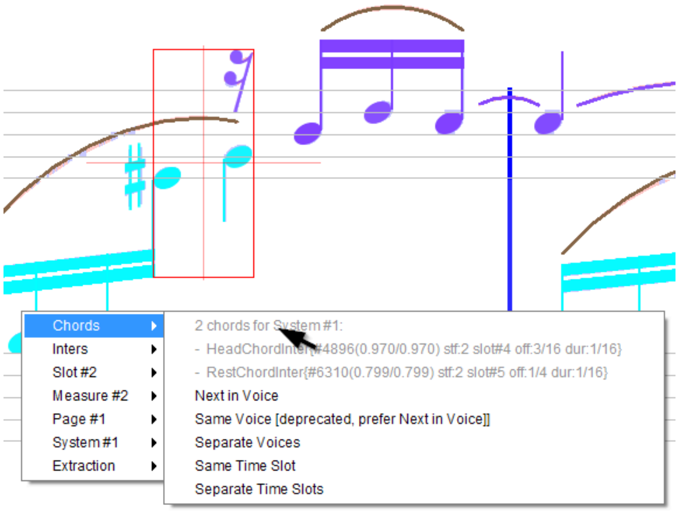
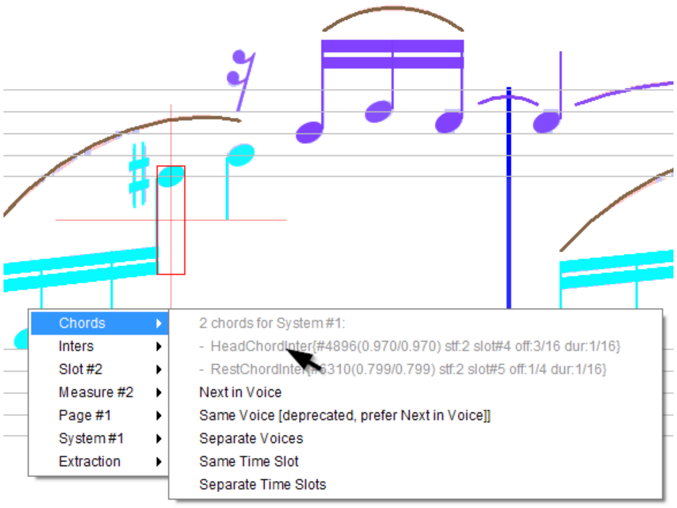


2 chords for System #1:

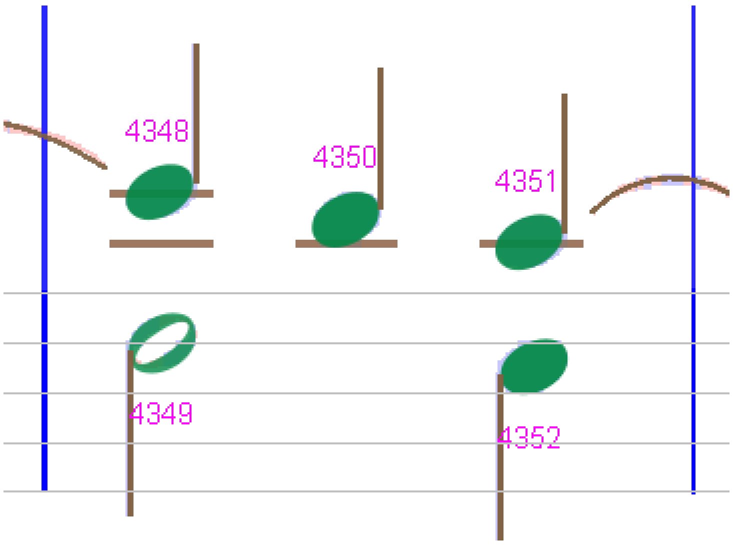
- HeadChordInter{#4896(0.970/0.970) stf:2 slot#4 off:3/16 dur:1/16}
- RestChordInter{#6310(0.799/0.799) stf:2 slot#5 off:1/4 dur:1/16}

Notice that the global bounding box (gray rectangle) encompasses the bounds of both chords.

Also, the menu begins with information lines about the selected chords. And if we hover over these lines, the bounding box is dynamically updated to show just the selected item. This is meant to allow a visual check of the selected chords:

all chords	first chord
 Chords Inters Slot #2 Measure #2 Page #1 System #1 Extraction 2 chords for System #1: - HeadChordInter{#4896(0.970/0.970) stf:2 slot#4 off:3/16 dur:1/16} - RestChordInter{#6310(0.799/0.799) stf:2 slot#5 off:1/4 dur:1/16} Next in Voice Same Voice [deprecated, prefer Next in Voice]] Separate Voices Same Time Slot Separate Time Slots	 Chords Inters Slot #2 Measure #2 Page #1 System #1 Extraction 2 chords for System #1: - HeadChordInter{#4896(0.970/0.970) stf:2 slot#4 off:3/16 dur:1/16} - RestChordInter{#6310(0.799/0.799) stf:2 slot#5 off:1/4 dur:1/16} Next in Voice Same Voice [deprecated, prefer Next in Voice]] Separate Voices Same Time Slot Separate Time Slots

To ease manual dealing with chords, we can make each chord ID visible as in the picture below (via the pull-down menu `View → Show chord IDs`):



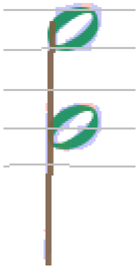
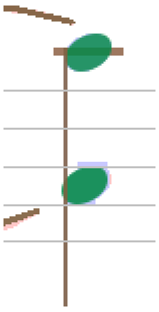
NOTE
The example Chords menu above shows only a partial list of possible chords actions, because the list depends on the current status and configuration of the selected chords.

Next, we list all the possible items of the `Chords...` menu.

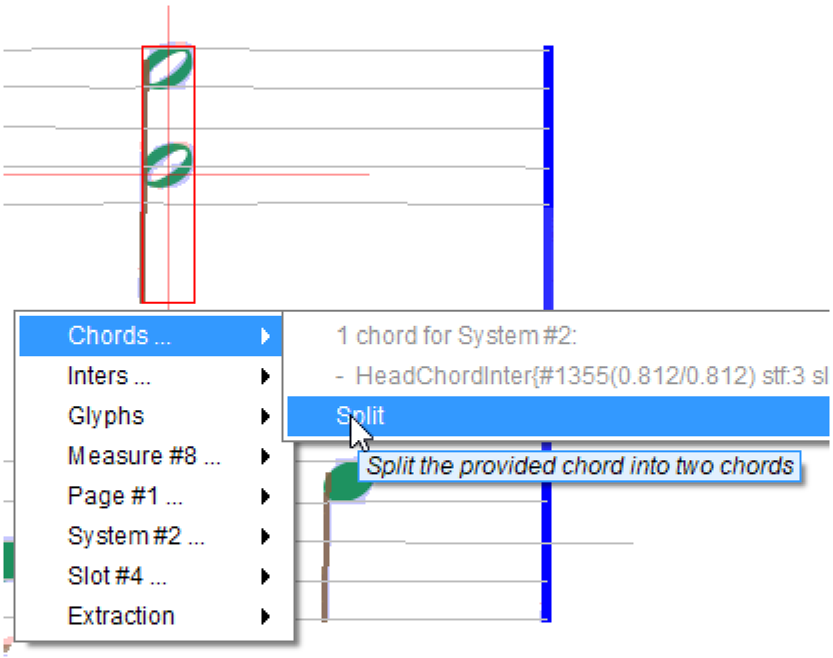
Chord

The gathering of note heads into chords may need some user correction.

Split

One chord?	Two chords?
	

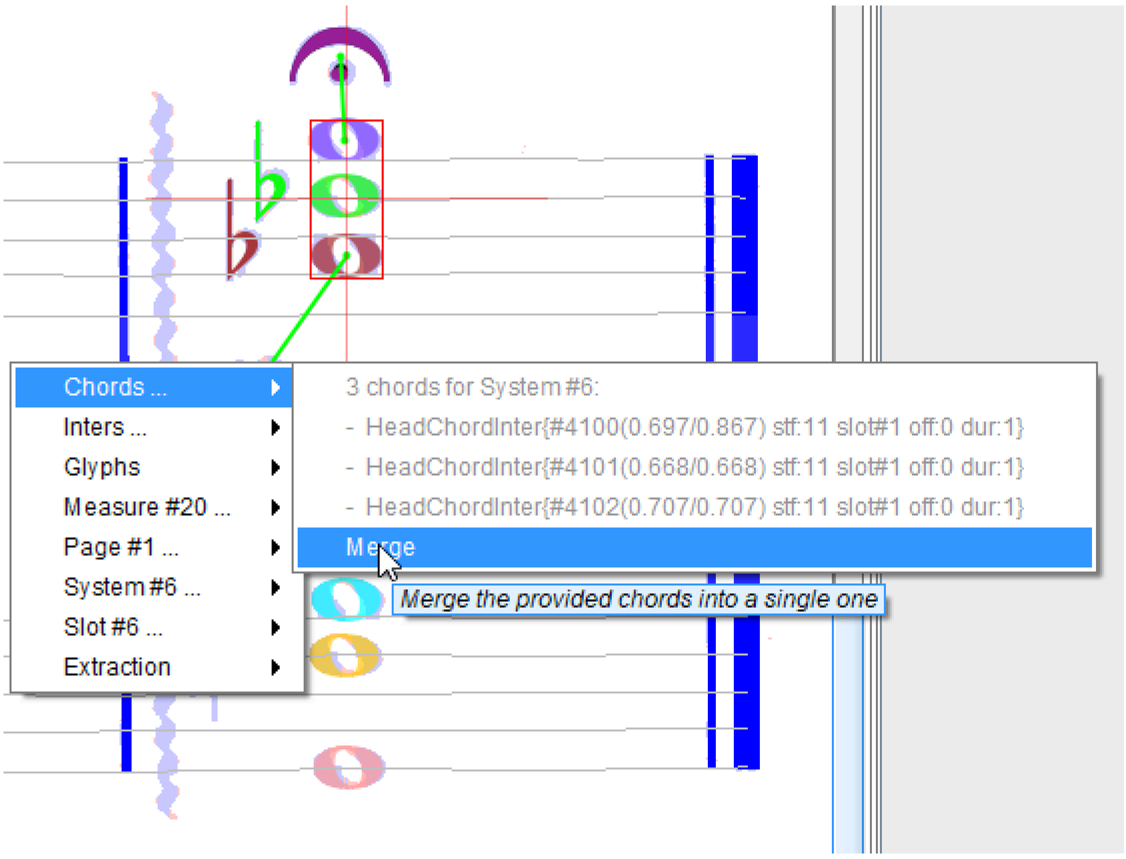
The OMR engine may have considered this as just one chord with a long stem, whereas we find these are in fact two separate chords, one above the other. In that case, select this chord and use the `Split` command.



Merge

Or, just the opposite, we may want to merge these two chords into a single one. In that case, we select both chords and use the `Merge` command.

In the specific case of whole notes, the `Merge` command is often needed. Because there is no stem involved, the engine has no clear heuristic [1] to gather whole heads into one chord.



Chords ...
Inters ...
Glyphs
Measure #20 ...
Page #1 ...
System #6 ...
Slot #6 ...
Extraction
3 chords for System #6:
- HeadChordInter{#4100(0.697/0.867) stf:11 slot#1 off:0 dur:1}
- HeadChordInter{#4101(0.668/0.668) stf:11 slot#1 off:0 dur:1}
- HeadChordInter{#4102(0.707/0.707) stf:11 slot#1 off:0 dur:1}
Merge
Merge the provided chords into a single one

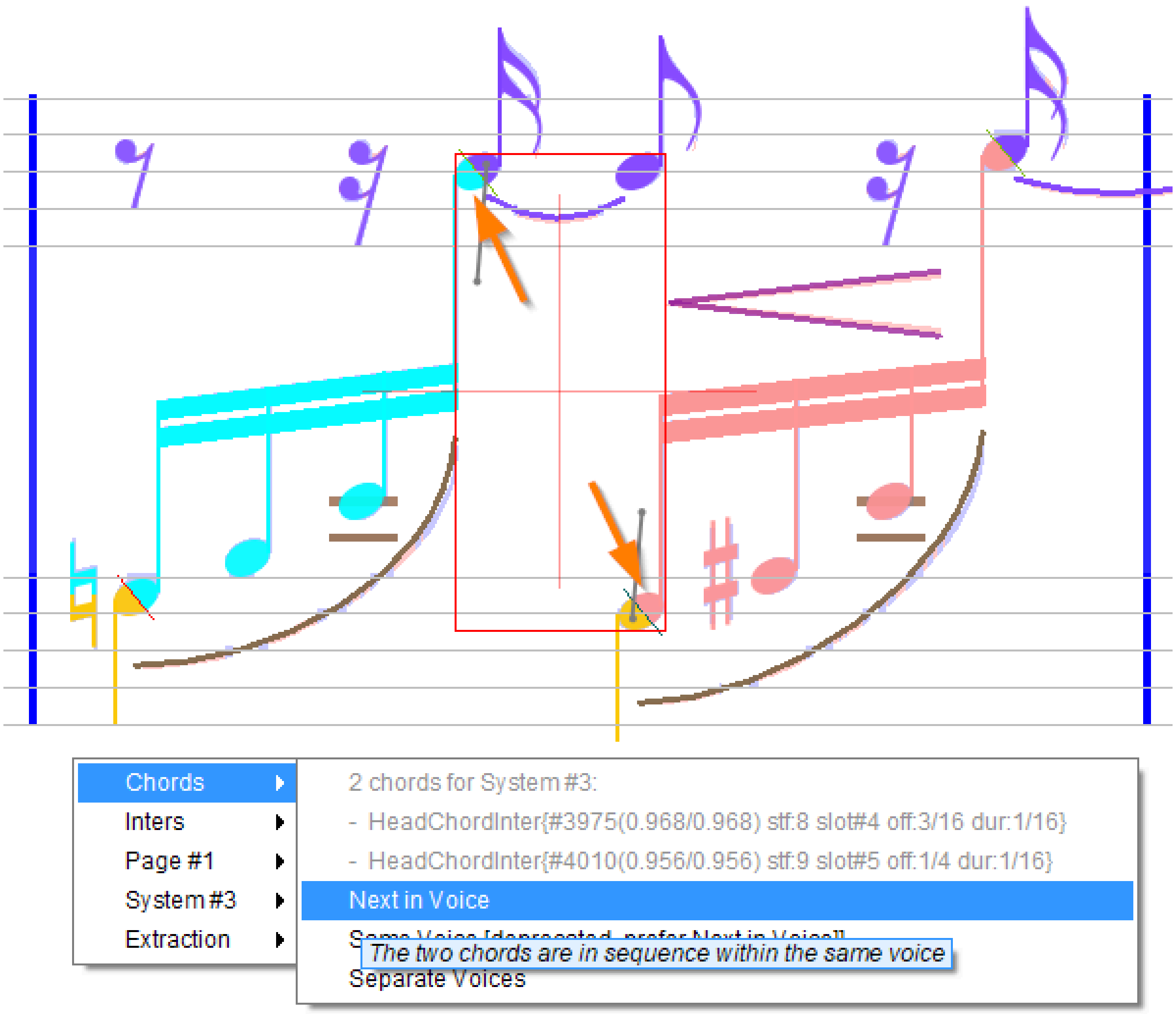
Voice

A voice is defined as a sequence of chords (head chords and rest chords) in the same music part. [2]

The Audiveris algorithm for voice building is already very tricky. It tries to reconcile different heuristics, but in some cases the result may not be the one the user would expect.

The purpose of these voice actions is to guide the engine in voice building.

Next in Voice



```
2 chords for System #3:
- HeadChordInter{#3975(0.968/0.968) stf:8 slot#4 off:3/16 dur:1/16}
- HeadChordInter{#4010(0.956/0.956) stf:9 slot#5 off:1/4 dur:1/16}
```

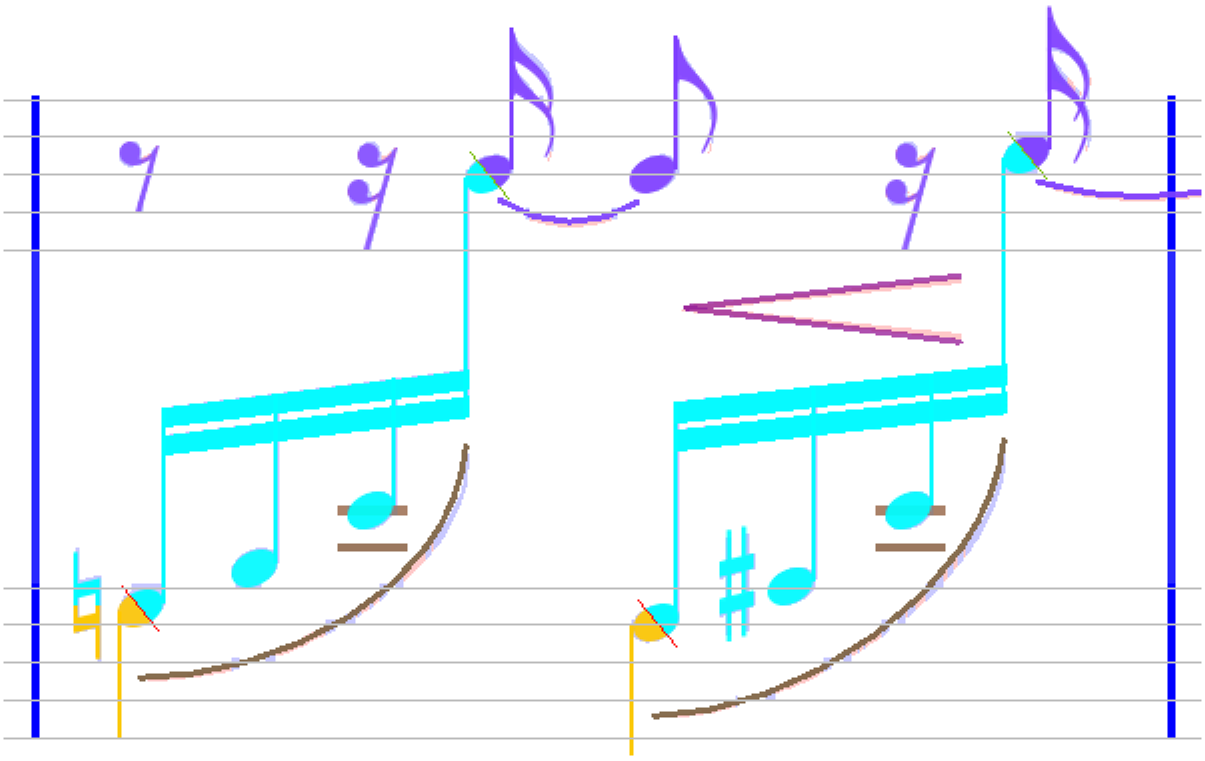
The two chords are in sequence within the same voice

The `Next in Voice` command operates on a horizontal sequence of 2 chords, say chord A and chord B, by establishing a relation from A to B, stating two things:

1. That voice of chord B should be dynamically **copied** from chord A.
   If later, for some reason, voice of chord A gets modified, the same modification will propagate to chord B.
2. And that chord B should start at (chord A start time + chord A duration).
   In other words, chord B time slot should **immediately** follow chord A time slot.

So we have a double dynamic propagation rule: on voice and on time. This is the reason why `Next in Voice` is now preferred to the old `Same Voice` command.

Here is the result:



We can always **undo** such an action, as any other UI action described here, via the `Sheet → Undo` or `Ctrl+Z` standard commands.

Also if we want, much later in the process, to cancel this task, we can always get back to selecting the same chords and we'll be offered to **cancel** the task. Cancelling this action removes the relation (and thus the related guidance).

Same Voice

Since 5.4 release, the old `Same Voice` command has been deprecated in favor of the more efficient `Next in Voice` command.

Separate Voices

This command imposes the voice algorithm to assign the selected chords to **separate** voices.

Note this is not exactly the reverse of the `Next in Voice` command (or the weaker `Same Voice` command):

- Without any command, we let the algorithm decide with no guidance.
- With a command (whether it's *next*, *same* or *separate*), we explicitly guide the algorithm.

Preferred Voice

Whereas the `Next in Voice`/`Same Voice` and `Separate Voices` commands operate on 2 chords, by establishing a **dynamic** computing rule from chord A to chord B, the `Preferred Voice` command operates on the single chord at hand, by assigning this chord a **fixed** voice numeric value.

This feature is effective only on the *first chord* in each measure voice.

It can be useful only for the very first chord of a voice in a system:

- Because the A-B relations can take place only within the same system (SIG), there is no way to establish a voice computing rule across systems.
- Specifically, we may need a way to set the voice number of a chord at the start of the very first measure of a movement.

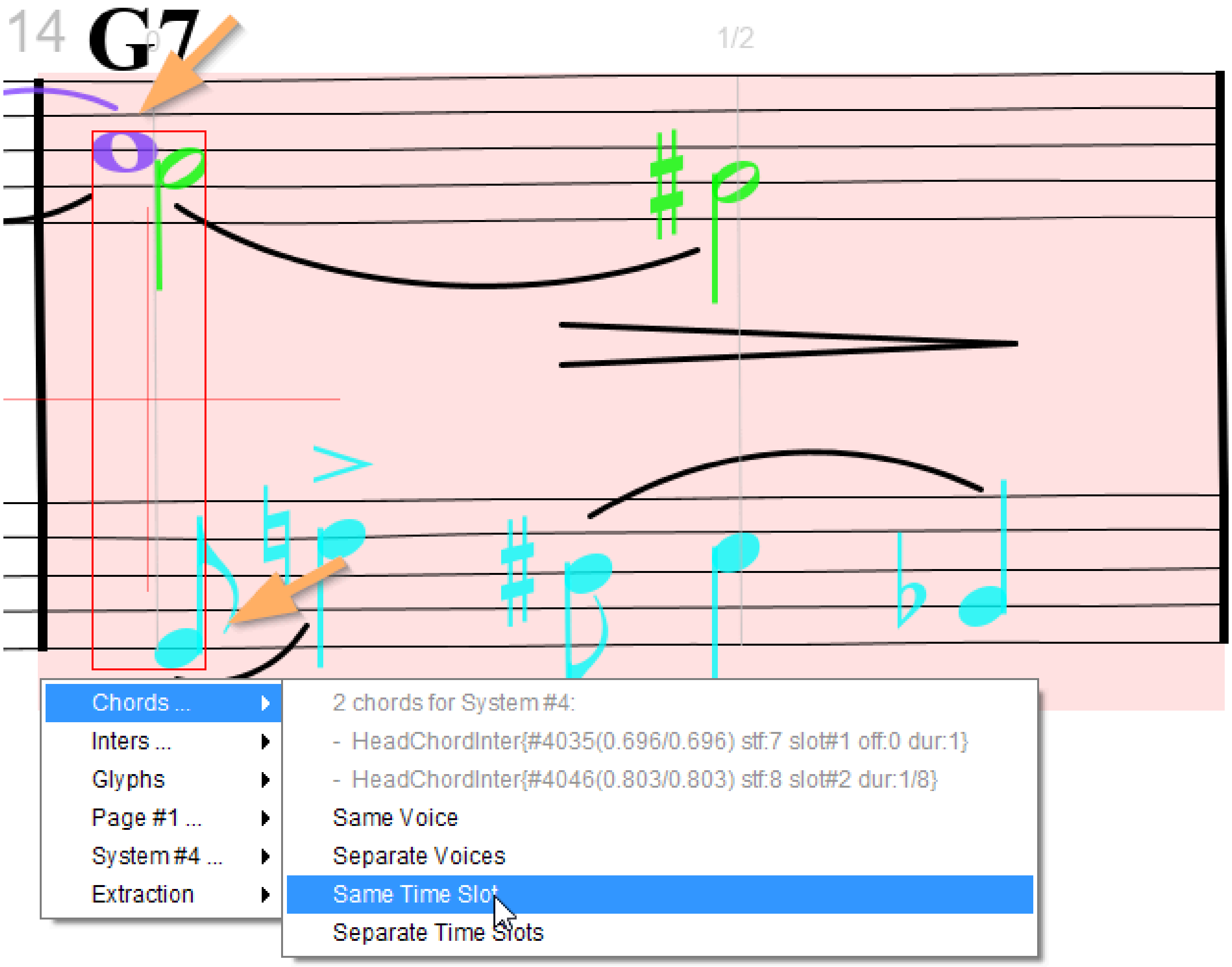
> These are the two cases where this feature can be useful.
> But except for these very specific cases, we are advised not to use this feature. In particular, let’s not imagine we could use it to somehow build voice sequences of chords!

Time

Assigning a chord to the proper time slot is as tricky as voice assignment. In fact, the time and voice algorithms are tightly coupled.

When two chords are rather close horizontally, when should we consider them as part of the same time slot?

Same Time Slot



Here, we can see that time slots on the second staff of the part are not correctly assigned. This is because the whole note on the upper staff and the 8th note on the lower staff are too far apart horizontally. So, we force these two notes to share the same time slot.

Experience shows that the most efficient action is generally to grab the set of *all* the chords that should share the same slot (a rather vertical selection **within the same part**) and apply the `Same Time Slot` command on the whole set.

Separate Time Slots

As opposed to `Same Time Slot`, this command is used to force time separation between two chords that the engine has considered as adjacent.

Footnotes

[1] The current heuristic for whole chords is to gather whole heads if they are aligned vertically and not more than one interline apart. ↩

[2]

There is an on-going debate about the possibility for Audiveris to share [rest-] chords between voices. But for the current 5.9 release, a chord can be assigned to exactly one voice. ↩

Text UPDATED FOR 5.6

The recognition of textual elements is delegated to the Tesseract OCR library.

The `TEXTS` step runs the OCR on the current sheet. We can also manually run the OCR on a selected collection of glyphs or even drag n’ drop text items from the `Shape` board.

Running the OCR results in one or several text words gathered in sentences, which we can further modify manually, in terms of:

- textual *content*,
- font *attributes* and *size*
- *type* of words and sentences,
- *role* of every sentence.

TABLE OF CONTENTS

TEXTS step

The `TEXTS` step runs the OCR on the whole sheet image and tries to assign to each OCR'd item its content, attributes, size, type and role.

This engine step is influenced by three options available in the `Book → Set book parameters` menu:

□ Support for chord names

☑ Support for lyrics (assumed to be located below the related staff)

□ Support for lyrics even located above staff

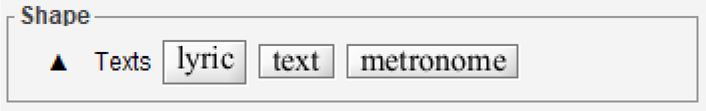
Chord names and lyrics are special items; this is the reason why their recognition must be explicitly selected to avoid collateral damages of the OMR engine when they are not desired.

On the other hand, the metronome marks, thanks to their recognizable structuring, don't require the setting of any specific option.

Manual OCR

The OCR can also be launched manually on a glyph(s) selection by pressing one of the buttons provided in the `Texts` palette of the Shape board:

- The `text` button
- The `lyric` button,
- The `metronome` button,



There are separate buttons because lyric items have a behavior significantly different from other text items – especially the gap between words can be much wider. And the metronome is a specific item on its own.

By manually choosing one button or another, we clearly specify the desired result type – and thus the sentence role – of the OCR operation.

We can as well drag n' drop items from the same `Texts` palette.
In this case, no OCR is performed, and we have to manually enter every word content.

Sentence vs. Words

A `Sentence` inter is an ensemble of one or several `Word` inter(s):

- Any `Word` handles its textual ***content***, font ***attributes***, font ***size*** and ***location***. We can manually modify any of these informations.

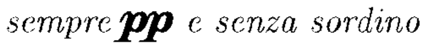
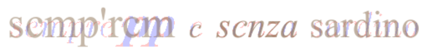
- A `Sentence` is a sequence of words (we can easily navigate from a selected word to its containing sentence via the `ToEnsemble` button of the InterBoard).

  A sentence is assigned a ***role***, which we can edit.

  The sentence ***content*** is simply defined as the concatenation of the contents of its words members. Except for the case of `Metronome` – which mixes text and music characters – we **cannot** modify a sentence content directly, but rather via each of its words members.

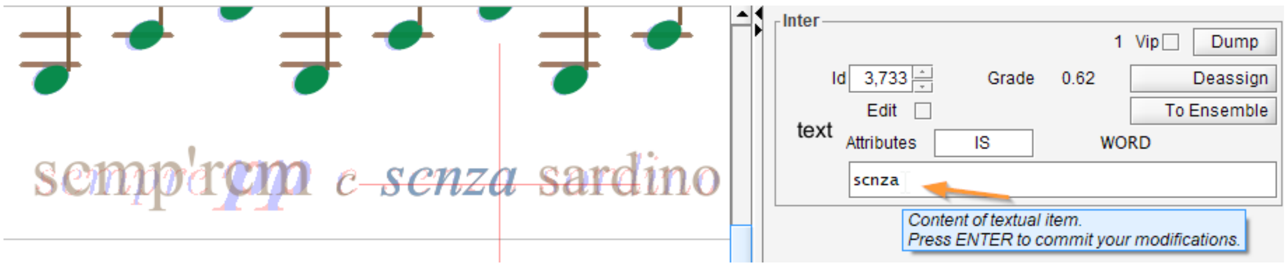
Word editing

Here is the example of an input line and the corresponding OCR result, at the end of the TEXTS step:

Source	Image
Input line	
OCR result	

Word content

The word “senza” has been OCR’d as “scnza”.



1 Vip □

Id 3,733 Grade 0.62

Attributes IS WORD

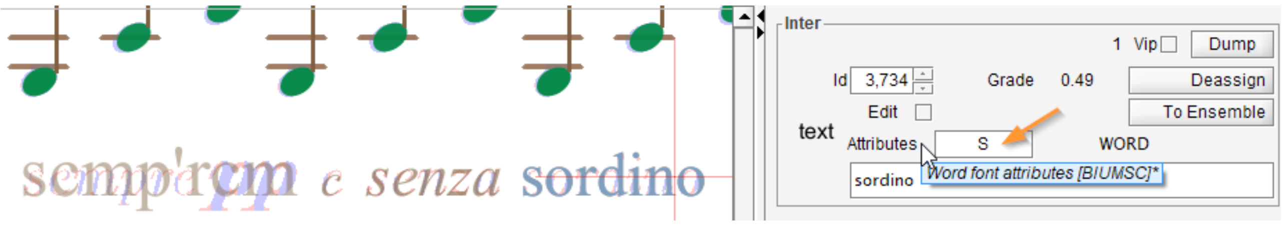
scnza

Content of textual item.
Press ENTER to commit your modifications.

We can modify the content in the text field and press `ENTER`.

Word attributes

We can do the same content modification for the word “sordino” OCR’d as “sardino”.



1 Vip □

Id 3,734 Grade 0.49

Edit □

Attributes S

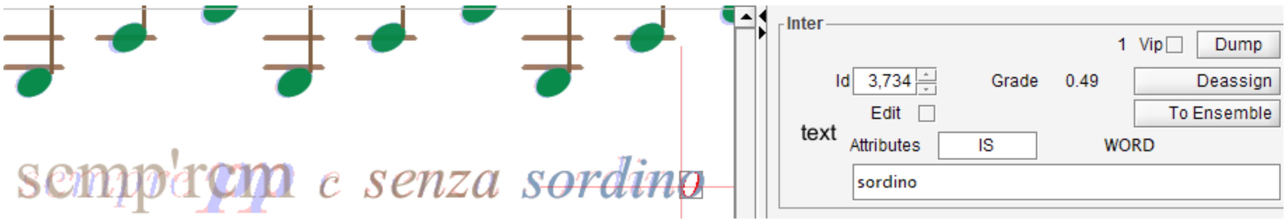
sordino

But this is not enough. The source was in italic and the result is displayed in upright style.

To fix this, we can now modify the word ***attributes***, via the `Attributes` field.

For “senza” the OCR’d attributes were “IS” (**I**talic **S**erif), while for “sordino” the OCR’d attributes are just “S” (**S**erif).

So we can simply change the string in the `Attributes` field to “IS”, and press `ENTER`.



1 Vip □

Id 3,734 Grade 0.49

Edit □

Attributes IS

sordino

Handling word attributes is a new feature provided by the 5.6 release.

The attributes, as transcribed by OCR or modified by the user, can be represented by a string composed of the `BIUMSC` characters.

Letter	Meaning	Use
B	**B**old	Style
I	**I**talic	Style
U	**U**nderlined	- ignored -
M	**M**onospaced	Font type
S	**S**erif	Font type
C	Small **C**aps	- ignored -

These attributes are used to:

1. Choose the text font (based on the `M` and `S` attributes):
   The chosen font is *Serif* (if specified), otherwise *Monospaced* (if specified), otherwise *Sans Serif* [1]

2 Apply a font style (based on the `B` and `I` attributes) : The chosen style is *Plain* or *Bold* or *Italic* or *Bold+Italic*.

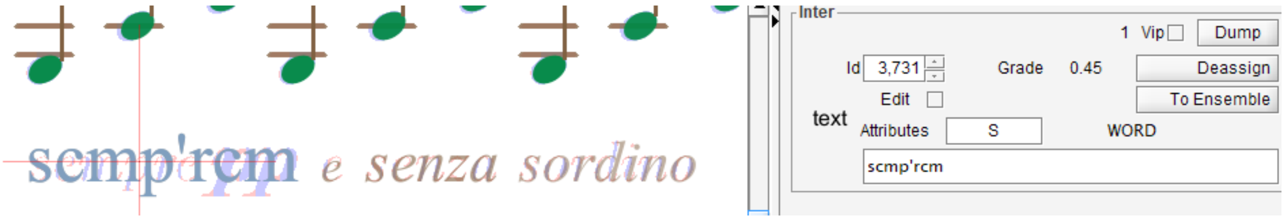
Note: As of this writing, the `U` (Underlined) and `C` (Small Caps) attributes are not supported and thus merely ignored.

This results in the possible combinations:

Type / Style	**( ) Plain**	**(B)old**	**(I)talic**
( ) Sans Serif	text	**text**	*tex*
(S)erif	text	**text**	*tex*
(M)onospaced	`text`	**`text`**	*`tex`*

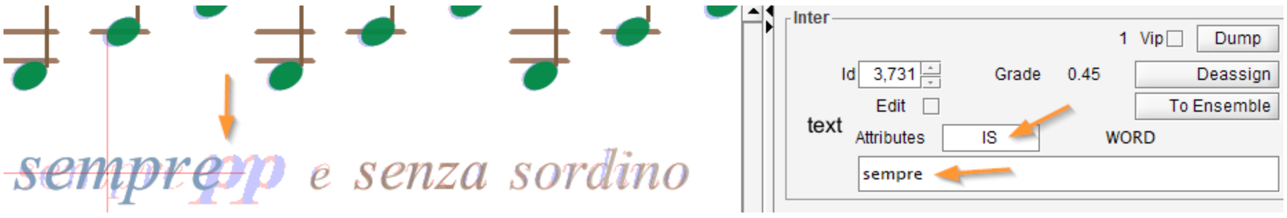
Word location and size

Still working on the same sentence, the very first word (“sempre”) needs to be fixed:



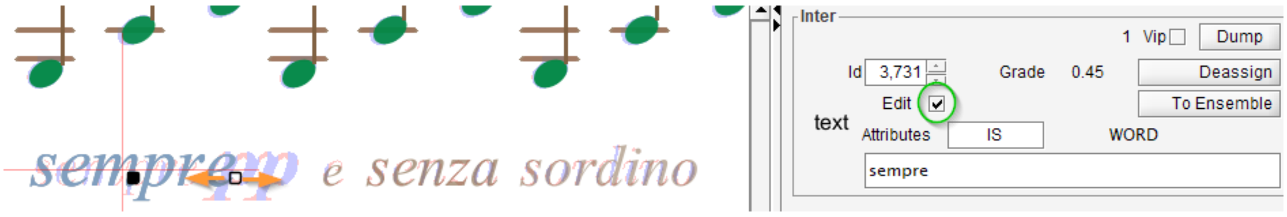
1 Vip □
Id 3,731 Grade 0.45
Edit □
text Attributes S WORD
scmp'rcm

Modifying the word content and attributes, we get this first result:



1 Vip □
Id 3,731 Grade 0.45
Edit □
text Attributes IS WORD
sempre

The word bounds are too wide when compared to the underlying pixels. To fix this, we have to put the word into the `edit` mode, either via a double-click on the word, or by ticking the `edit` checkbox in the InterBoard.



Id 3,731

Grade 0.45

Edit ☑

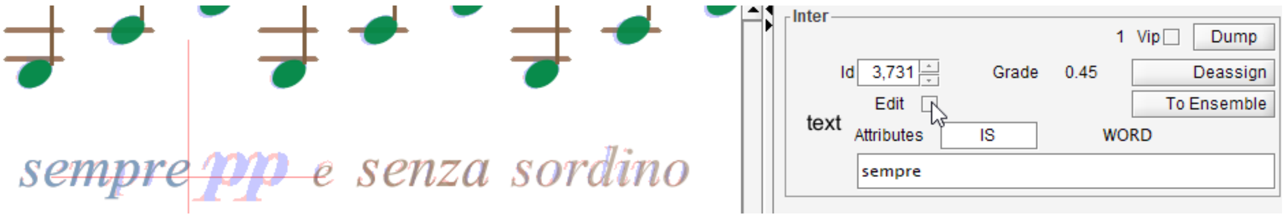
Attributes IS

sempre

A word being edited shows 2 handles:

- The *middle* handle can move the word into any direction
- The *right* handle can increase or decrease the word size (and the related font size accordingly).

Moving the right handle to the left allows to reduce the word width:



Id 3,731

Grade 0.45

Edit □

Attributes IS

sempre

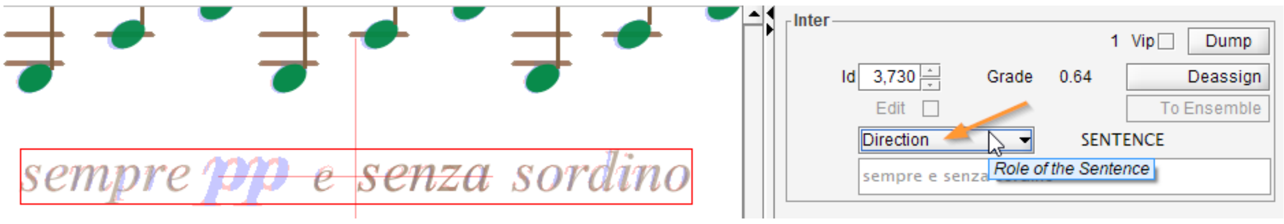
And later, in the `SYMBOLS` step, the “PP” glyph will indeed be recognized as a pianissimo symbol.

Sentence editing

A sentence role can be set to any value among:

- UnknownRole
- ***Lyrics***
- ***ChordName***
- Title
- Direction
- Number
- PartName
- Creator
- CreatorArranger
- CreatorComposer
- CreatorLyricist
- Rights
- EndingNumber
- EndingText
- ***Metronome***

Since the 5.2 release, in all cases, we can manually modify the sentence role afterwards, from any role to any other role.



Id 3,730 Grade 0.64

Direction SENTENCE

sempre e senza

Plain sentence

A “plain” sentence is any sentence which is assigned a role different from `Lyrics`, `ChordName` and `Metronome`.

Following an OCR recognition (`Texts` step or manual OCR), the role of each resulting *plain* sentence is precised. Based on a bunch of heuristics, the engine tries to further distinguish between plain roles like: direction, part name , title, composer, lyricist, etc.

Chord name

A chord name is a musical symbol which names and describes the related chord.

For example: `C`, `D7`, `F♯`, `B♭min`, `Em♭5`, `G6/B`, `Gdim`, `F♯m7`, `Em♭5`, `D7♯5`, `Am7♭5`, `A(9)`, `BMaj7/D♯`.

> **IMPORTANT**
> As of this writing, the Audiveris engine is not yet able to recognize chord names that include true sharp (`♯`) or flat (`♭`) characters. Perhaps one day, we will succeed in training Tesseract OCR on this text content.
> For the time being, Audiveris is able to recognize such chord names when these characters have been replaced (by OCR “mistake”, or by manual modification) by more “usual” characters:
> - `'#'` (number) as replacement for `'♯'` (sharp),
> - `'b'` (lowercase b) as replacement for `'♭'` (flat).

When we OCR a chord name word, Audiveris may be able to decode it as a chord name and thus wrap it within a chord name sentence.

If Audiveris has failed, we can still force the chord name role (at the sentence level) and type in the missing `b` or `#` characters if so needed (at the word level). The chord name will then be decoded on-the-fly with its new textual content.

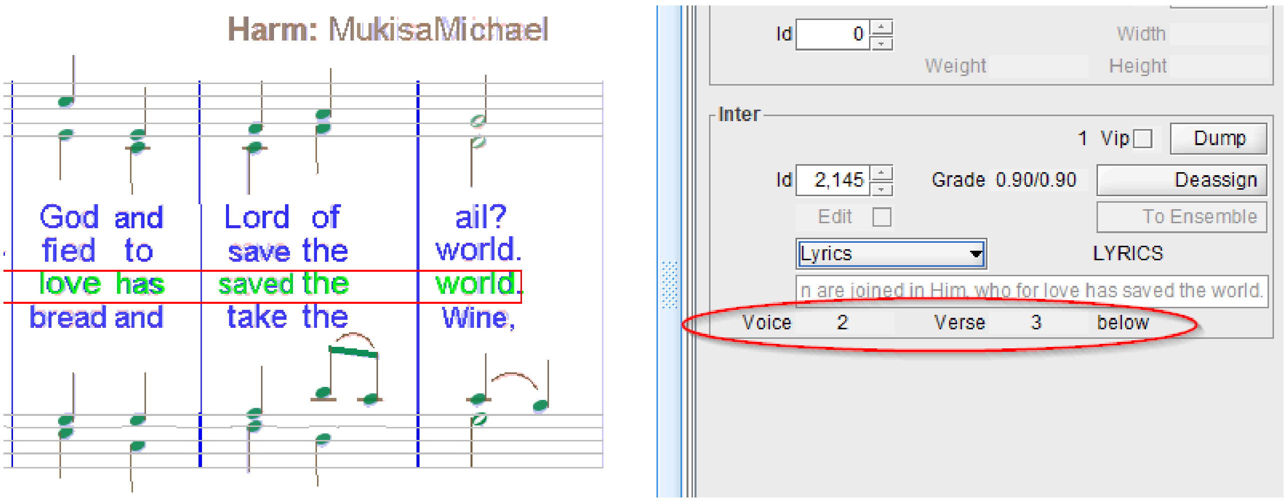
Note we don't have to manually enter the true sharp or flat signs. Entering them via their Unicode value is a bit tricky and, in the end, useless.
Instead, when text has been recognized or assigned as a chord name, its internal `b` or `#` characters are automatically replaced by their true alteration signs.
For example, we can type "Bb" then press `Enter` and the chord name will be translated and displayed as "B♭".

Lyric line

A lyric line is a sentence composed of lyric items.

When selected, the `Inter` board displays additional data:

- Voice number,
- Verse number,
- Location with respect to staff.



Id 0
Width
Weight
Height
Inter
1 Vip □ Dump
Id 2,145 Grade 0.90/0.90 Deassign
Edit □ To Ensemble
Lyrics LYRICS
n are joined in Him, who for love has saved the world.
Voice 2 Verse 3 below

Each syllable (lyric item) is usually linked to a related chord, either above or below.
But it is not always obvious whether the text concerns the staff above or below nor is it always clear which voice is concerned.

If a syllable is not linked to the correct chord, we can modify this link manually by dragging from the syllable to the suitable chord. This will update on-the-fly the line data (voice, verse, location).

Metronome mark

Since the 5.4 release, the metronome marks can be automatically recognized.
We can also edit them afterwards and even create new marks from scratch.

A metronome mark is a sentence composed of words.
One of its words is special as it contains not textual characters but music

characters. This word is the `BeatUnit` word.

Editing the metronome is detailed in this specific section.

Footnotes

[1] Combining `M` (monospaced) and `S` (serif) could produce a font like “Courier New,” but the result is unsatisfactory. Therefore, in “MS” the `M` is ignored in favor of the `S`. ↩

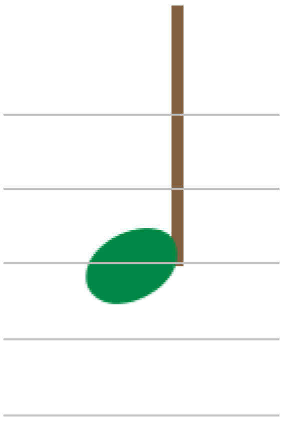
Shared Head

TABLE OF CONTENTS

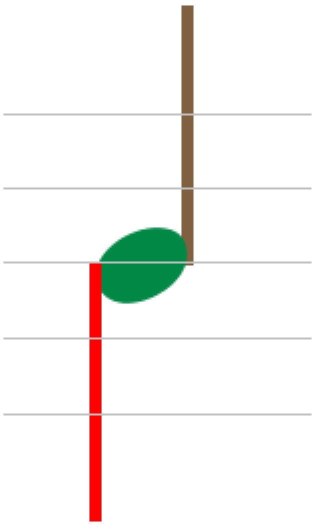
Note head shared by two voices

Generally a note head (filled or void – except whole and breve notes) is connected to exactly one stem, as in the following example.

If head and stem are connected (via a `HeadStemRelation`), both appear with their own standard color. If not, one or both appear in red and to fix this, we can simply drag a link from one to the other.



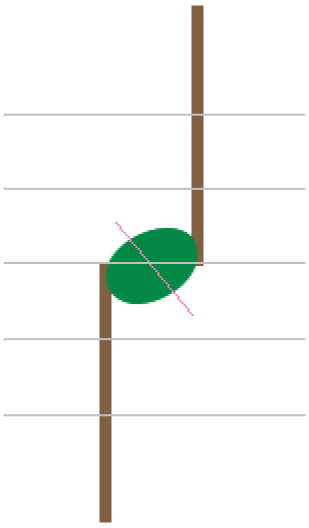
Then, if needed, we can insert a stem on the other side of the head, in the opposite direction.



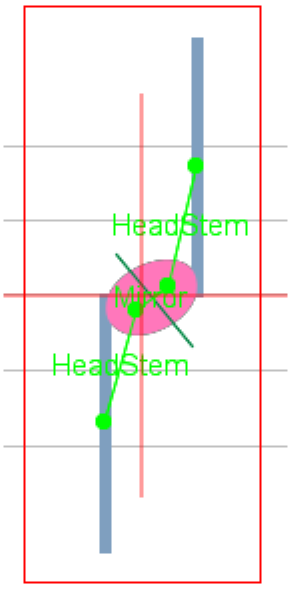
It appears in red because it can’t get automatically connected to the head (because this head is already connected on the other side).

To actually set the connection, we have to **manually** drag a link between the (new) stem and the head. The program then checks if the resulting configuration is the canonical one (which means *a stem down on the left side of the head and a stem up on the right side, the shared head being located at the end of each stem*):

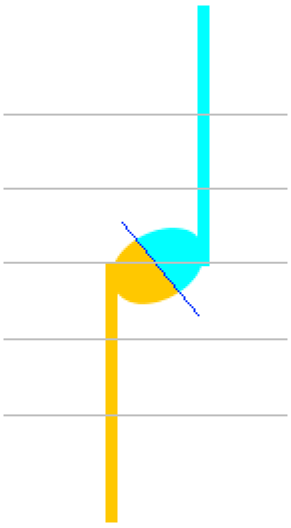
- If the check fails, the new connection is set but the old one is removed. This is the standard behavior.
- But if the check succeeds, both connections are kept and the *‘shared’* head gets logically duplicated into two heads, one *‘half’* for the left and one *‘half’* for the right:



By selecting all components, we can see the various links (*HeadStemRelation* between each head half and “its” stem, and *MirrorRelation* between the two half heads):

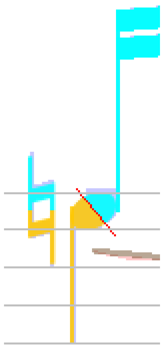
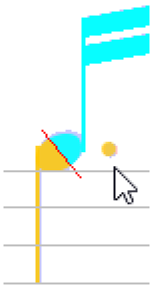


And if voices are colorized, the separation between head ‘halves’ becomes even more visible:



Impact on relations

Playing with relations around note heads, such as the relation with an accidental or with an augmentation dot, is still possible with shared note heads. Simply, we have to pay attention to point precisely to the desired *‘half’* head.

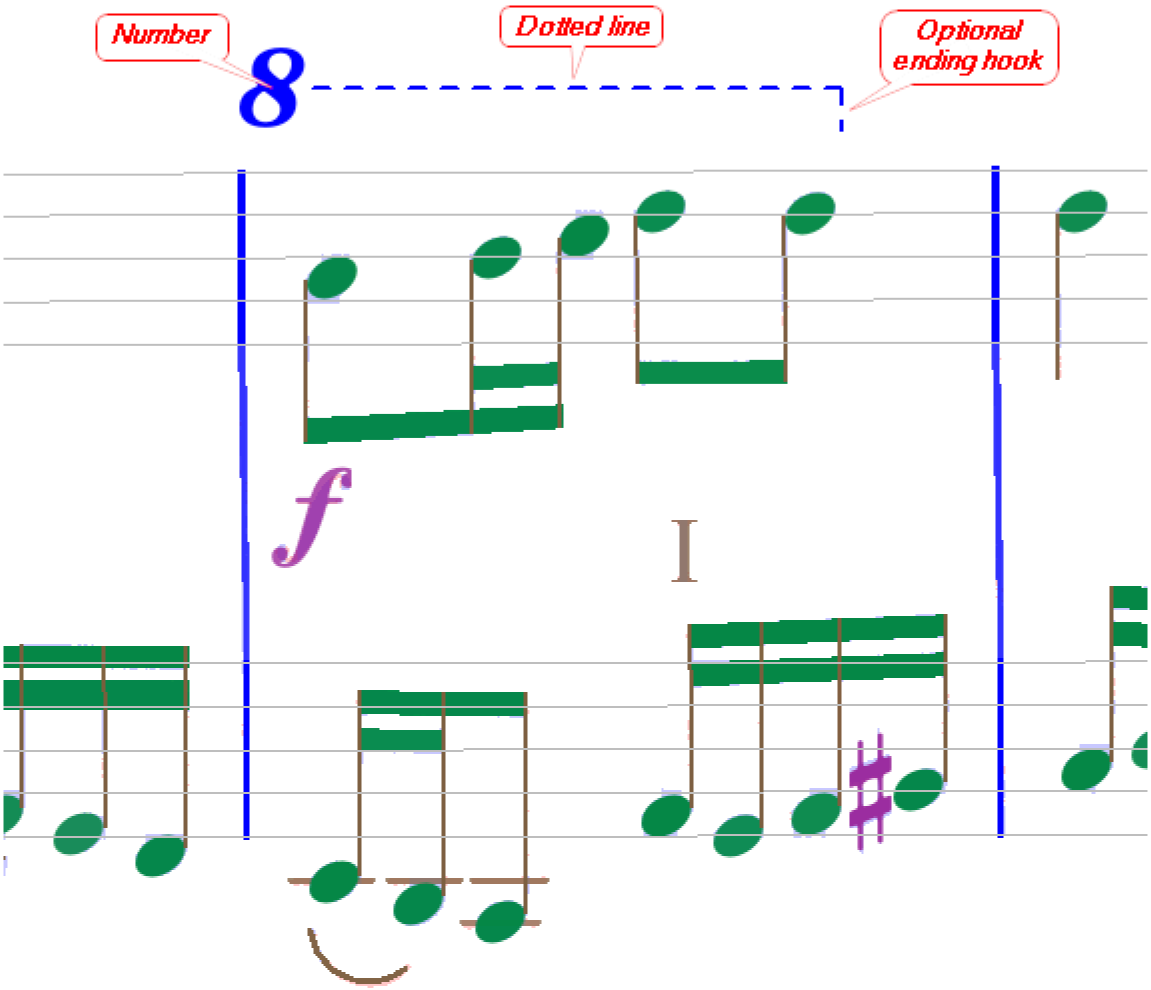
Example	Explanation
	Here, the alteration sign is *‘shared’* and thus also split, each sign *‘half’* colorized with the same color as its related head half.
	Here, the augmentation dot is related only to one head *‘half’* (otherwise the dot would exhibit both colors as the alteration sign in the previous example)

Octave shift

TABLE OF CONTENTS

Definition

An octave shift looks like this:



Its purpose is to tell a reader that there is a local shift in height between the notes as they are written and as they must be performed.

- The ***number*** on left side can be `8` (*Ottava*), `15` (*Quindicesima*) or `22` (*Ventiduesima*) for a 1-, 2- or 3-octave shift respectively.
- The horizontal straight ***dotted line*** recalls the staff horizontal range where the shift applies.
- The ending ***hook***, when present, gives a precise end for the shift, and generally points back to the related staff.
- The shift ***kind*** (`alta` or `bassa`) tells whether the notes must be performed respectively higher or lower than written.

Unfortunately the shift *kind* is not easy to detect in a printed score:

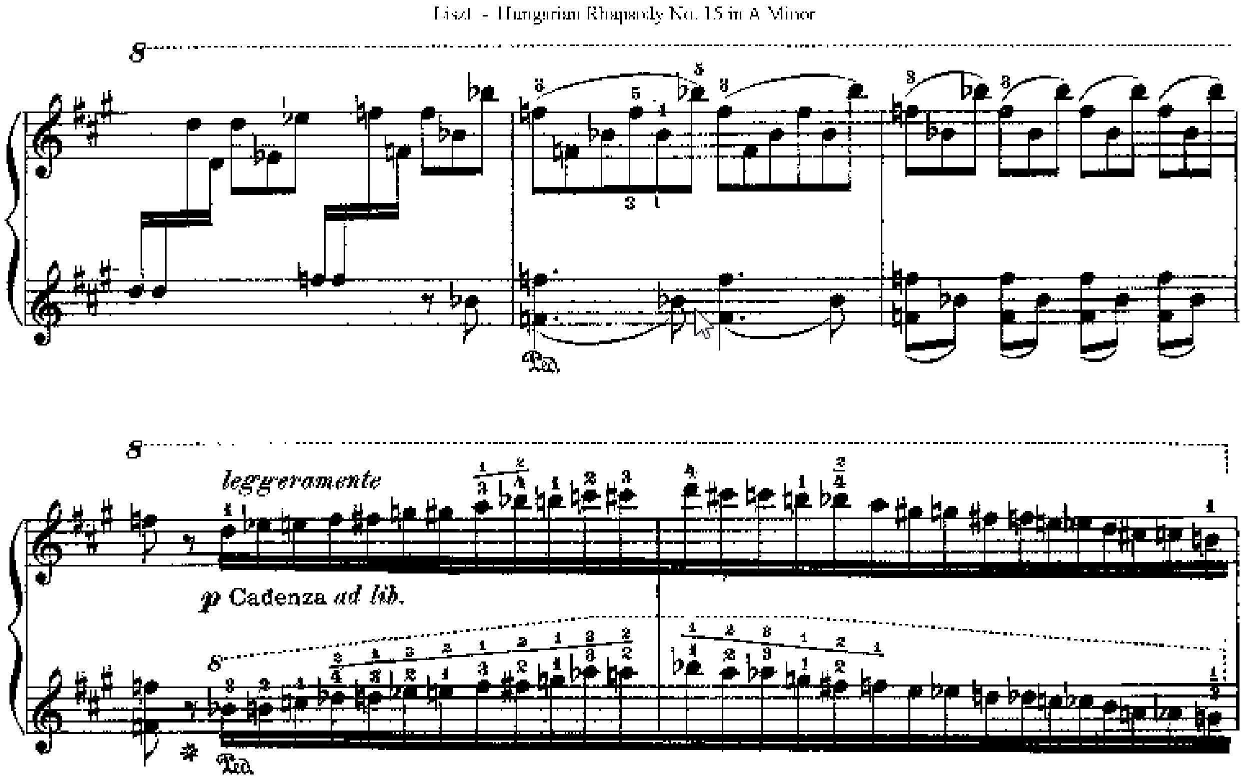
- Some scores exhibit various glyphs, such as `8va`, `8vb`, `8va alta`, `8va bassa`, etc to name a few ones related to ottava. Similar variations can be found for quindicesima and for ventiduesima.
- Sometimes the `va` suffix, when present, appears at the top or at the bottom of the `8` glyph.
- Sometimes the vertical location of the sign with respect to the related staff is used (above the staff to indicate `alta`, below the staff to indicate `bassa`).

- And how can the OMR engine determine the related staff of an octave shift? Using the vertical distance to the closest staff is not reliable enough.

Multi-system octave shift

The example at the beginning of this page presented a rather short octave shift, limited in range to a portion of the physical staff just below it.

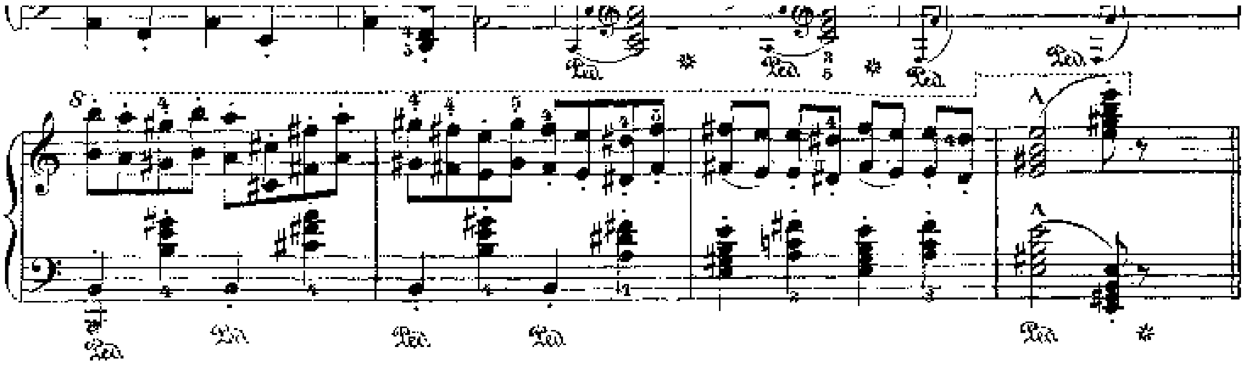
The following example is different:



We can see an octave shift starting on the upper staff in the first system and stopping at the end of the upper staff in the second system.

- Audiveris refers to this multi-system configuration as one *logical* octave shift composed of several *physical* octave shifts.
- Notice there is no ending hook for the first physical shift, indicating its continuation beyond the containing system.

Notice also the other shift located on the lower staff of the second system, or the next example below, and how the notion of “straight horizontal line” should be taken with a grain of salt…



Current status

As of this writing, automatic recognition of octave shifts by the Audiveris engine is very limited:

- The starting glyph (`8`, `15` or `22`) could be recognized,
- Potential suffixes much less easily,
- The dotted line and hook cannot be recognized by the current engine.

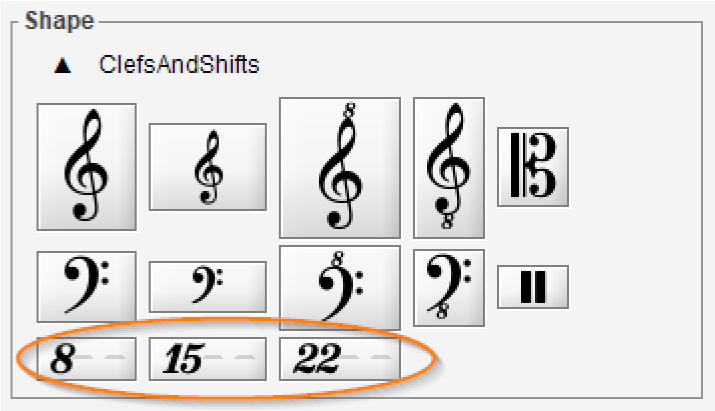
Instead, we have chosen to develop friendly means to let the user manually create and edit even complex octave shifts.

> **WARNING**
> The Audiveris implementation for octave shifts is made **under the following simplifications**:

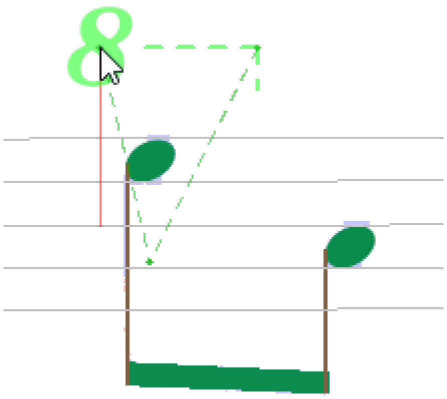
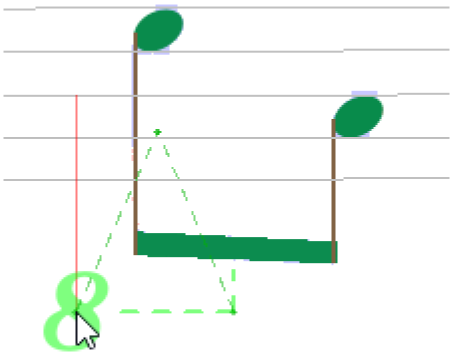
1. The user must explicitly indicate the related staff, in one way or another.
2. The octave shift number is limited to `8`, `15` or `22` glyph, with no suffix at all.
3. The octave shift kind is determined by its vertical location with respect to the staff, above for `alta`, below for `bassa`.
4. Logical octave shifts can cross system breaks but not page breaks.
5. A hook is always present at the end of the last physical octave shift, and only there.

Creation

The easiest way to create an octave shift is a drag & drop from the `ClefsAndShifts` family in the Shape board:



1. We drag the desired shift symbol
2. We pay attention to hover over the target staff.
   Staff information is sticky:
   - The symbol is initially displayed in gray with no hook, then in green when hovering over a staff
   - The last hovered staff becomes the current staff, and potential links to staff chords are shown
   - The symbol changes when being moved around the staff:

Above staff	Below staff
an `alta` downside hook appears	a `bassa` upside hook appears
	

3. We drop the symbol at the desired location (typically the center of the number box) by releasing the mouse.

Another way is to select a suitable glyph, which must be limited to the number part without any suffix:

1. This will trigger the glyph classifier which may recognize the shape, in which

case we just press the proper button in the classifier board.

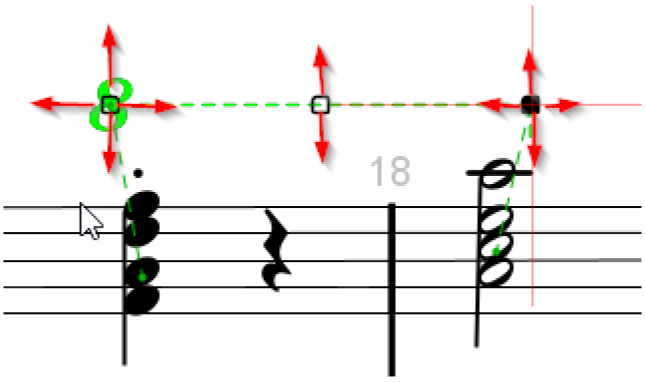
2. If not, we can manually assign the target shape via the pop-up menu or via a double-click in the `ClefsAndShifts` family of the Shape board.
3. Note that we will be prompted for the precise target staff of the created shift.

In both ways, what we have created is just a small octave shift. We can now proceed to its precise editing as described in the next section.

Editing

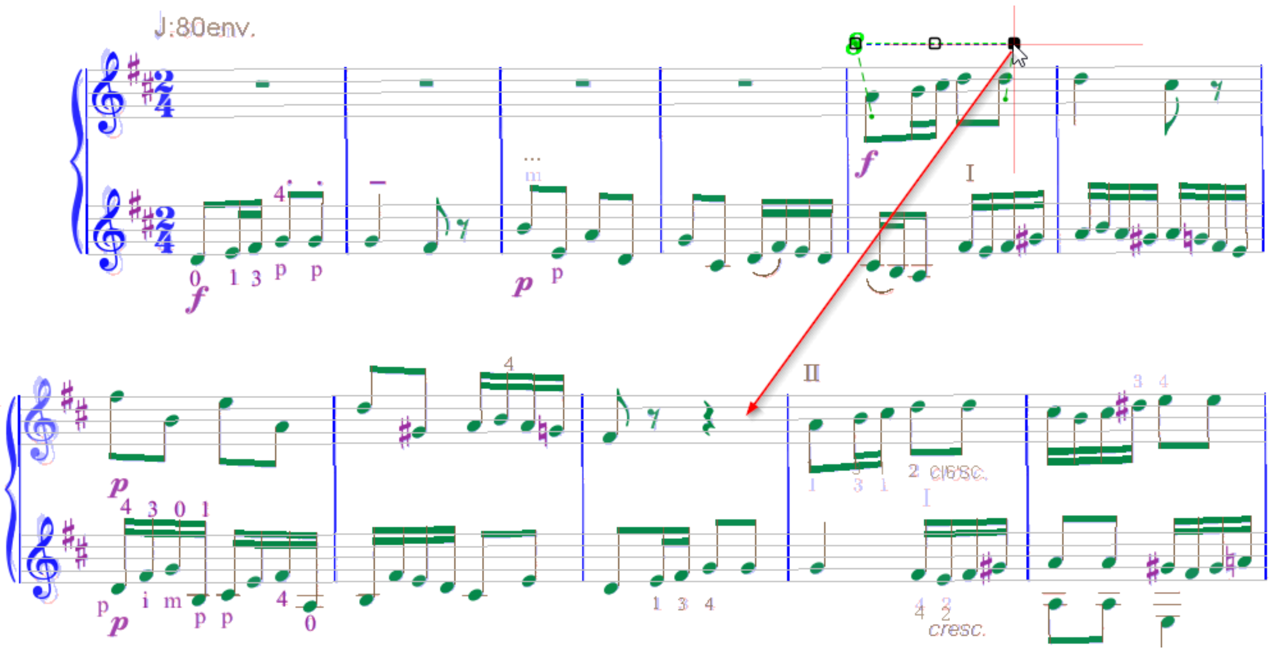
The octave shift editor provides up to 3 handles: left, middle and right:

- All these handles can move the shift vertically between the current staff and the other staff, above or below, according to the shift kind alta/bassa.
- Only the left and right handles, when present, can resize the shift line horizontally within current staff limits.

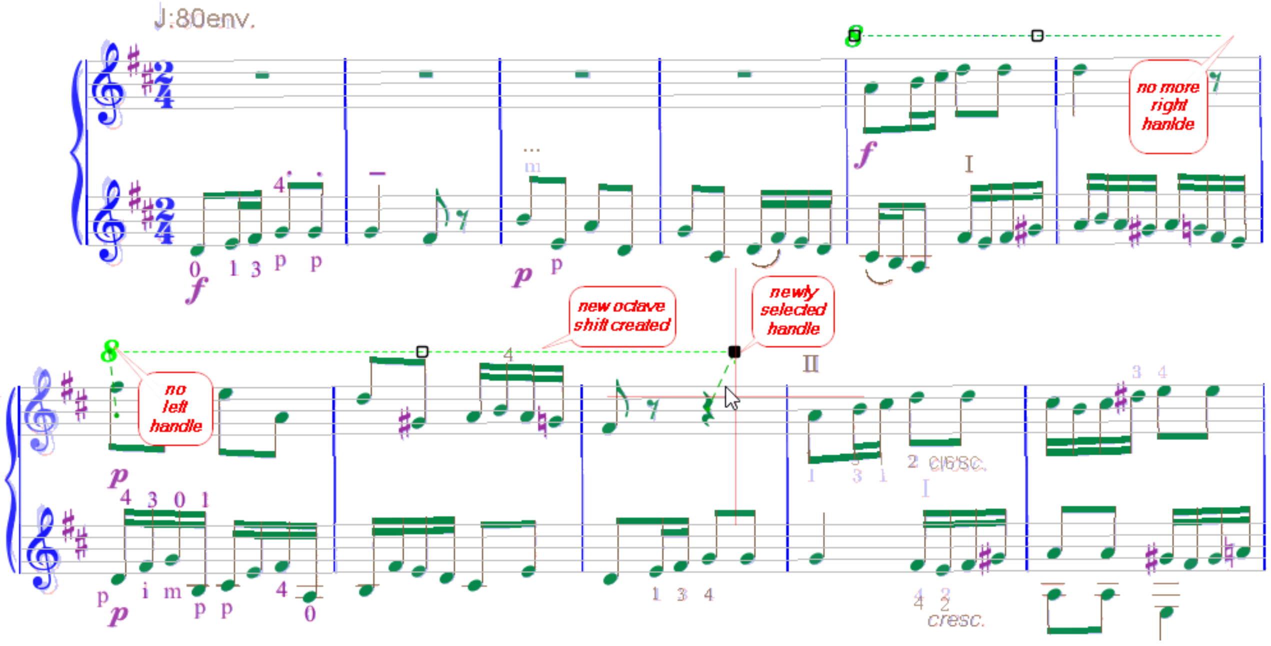


What is specific to this editor is the ability to drag the user location (shown as a red cross), way beyond the limits of the current system, until the corresponding staff in some previous or succeeding system.

In the example below, the user has selected the right handle as shown and will drag the mouse from staff #1 down to a location in staff #3:



When reaching this staff #3 (which is the succeeding staff for staff #1), the editor configuration changes on-the-fly:



Instead of the initial simple octave shift with 3 handles, we now have 2 physical shifts in a logical shift:

- The upper shift has been extended until the end of its staff, and its right handle removed.
- The lower shift has been created from the start of its staff (with no left handle) until the user release point.

The user can move further (though staying on the same page), extending the same logical shift with additional physical shifts – or conversely shrinking the logical shift by moving backwards.
Similar actions are possible, starting from the upper left handle, to extend the logical shift upwards.
Intermediate physical shifts can also be refined vertically.

Finally, clicking outside of any handle ends the current editing.

> **NOTE**
> A logical shift, whether it is made of one or several physical shifts, is managed as one entity. This applies to creation, editing and removal actions.

And as usual, any of these actions can be undone and redone at will.

Removal

The manual removal of any (physical) octave shift means the removal of the whole “logical” octave shift.

Tips and tricks

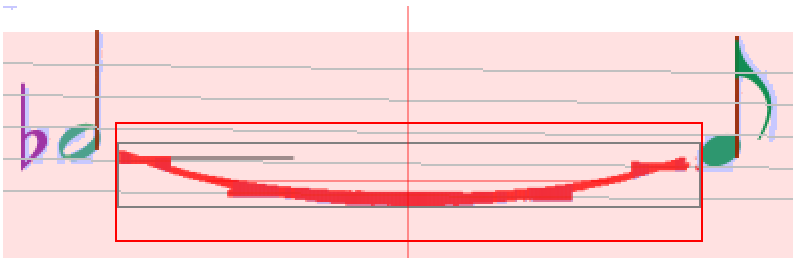
This section gathers a few things based on concrete user experience. These little tricks should save us time and efforts in our editing sessions.

TABLE OF CONTENTS

Select glyphs using the selection Frame

In bad scans, often elements are detected as two or more separate glyphs – this is rather frequent for slurs.

In such a case, we use a selection frame and try to catch all parts of a suitable glyph before we define an inter for it:



We have to make sure not to select parts of other elements (e.g. augmentation dots)!

Delete incorrect interpretations before defining new ones

Before assigning another interpretation to a glyph / a group of glyphs, we make sure to delete the previous one first. Otherwise, there will be two confusing interpretations in the output file.

> **NOTE**
> Audiveris is able to detect when the *exact same* glyph has been assigned two different interpretations, and thus remove the old one to just keep the new one.
> But glyphs that differ by one pixel or more are, by definition, different glyphs!
> So, when in doubt, we should not hesitate to clean up the scene beforehand.

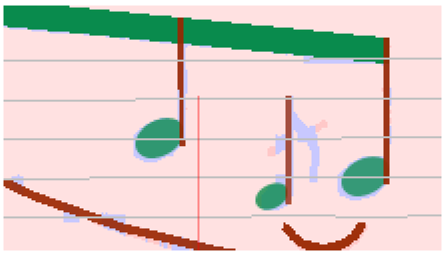
Look for missing / wrongly defined augmentation dots

The most frequent reasons for errors in the rhythm check are missing or wrongly detected augmentation dots.

So when all obvious reasons for incorrect rhythm are solved and the measure is still in pink, we can zoom into the image and look for augmentation dots in the concerned measure.

Wrongly detected grace notes

Sometimes grace notes are considered as standard notes with just a rather small head.
And while all standard notes are involved in rhythm building, true grace notes are not, because they are considered as mere ornaments.

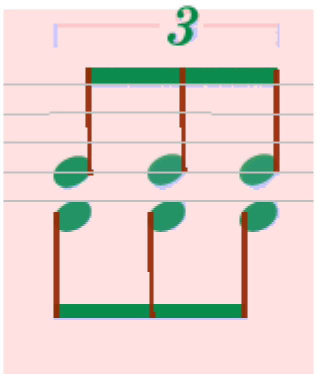
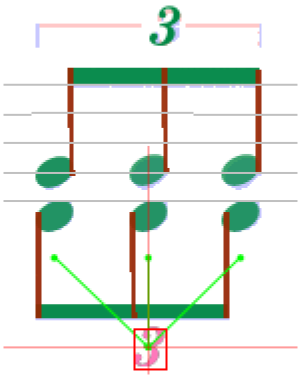


In such a case we just delete the note and re-define with the correct interpretation (from the `Ornaments` palette in Shape board)

Add triplet to 2nd voice with missing “3”

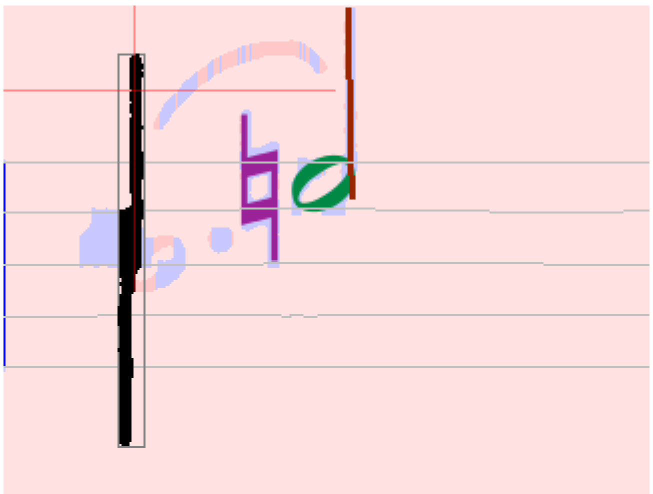
Sometimes scores that contain two voices use a “common” 3 for a tuplet in both voices.

In such a case we can add the missing triplet element by drag & drop to the 2nd voice (during dragging, a fine red line shows the related staff):

Lower triplet missing		Lower triplet added
	===>	

Perfectly opposite notes

Sometimes there are two notes of two voices with opposite stems in the same horizontal position.



In such a case Audiveris often does not detect the two stems as separate elements, but as one long stem (on which the heads candidates will appear in abnormal positions, and thus be ultimately discarded by the engine).

The easiest way to fix this is to directly insert two opposite *compound notes*. In the example at hand:

- Top: a quarter note with stem up,
- Bottom: a half note with stem down (complemented with an augmentation dot).

UI examples AUGMENTED IN 5.4

The purpose of this chapter is to present editing sessions on representative scores to be used as concrete examples for the Audiveris user.

We can expect this chapter to grow incrementally, as more and more example sessions get included here.

TABLE OF CONTENTS

O lux beata Trinitas

NOTE

You will notice that the description of user editing is very detailed in this example, perhaps too much. This was done on purpose, so that the session content could be used as an introductory material.

The case at hand describes the processing of an old office hymn available on the IMSLP site. You can download it directly via the button below:

Input file available here

The case originated from issue #481 posted on the Audiveris forum, which led to several discussions and engine improvements, and finally its integration as a UI example into the Audiveris handbook.

The main score specificities:

- Book of 3 movements over 5 sheets
- Non-standard time signature
- Whole rests that are not measure-long rests
- Voice sequences of whole note heads

The main UI actions:

- Use of “partial whole rests” option
- Common-cut (2/2) replaced by 4/2 time signature
- Missing half notes and whole notes
- Chords with no time offset
- Voice alignment

TABLE OF CONTENTS

Program launch

There are many ways to install/build then launch the Audiveris program, depending on our environment.

If we haven’t done so, we select Audiveris releases on github to retrieve the latest release.

Under Windows, we can simply run the installer (for example Audiveris_Setup-5.2.2-windows-x86_64.exe). Under Windows, Linux and macOS, we can decide to clone the Audiveris project locally and then build it. In any case, we will have to make sure that a proper Java version is installed.

Here are the first messages displayed, when pressing the Windows start icon, selecting the Audiveris folder, then the Audiveris program:

```
Audiveris version 5.2.2

LogUtil. Property logback.configurationFile not defined, skipped.
LogUtil. Configuration found C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\config\logback.x
LogUtil. Logging to C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\log\20210710T121739.log

Loaded plugins from C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\config\plugins.xml
Classifier loaded XML norms.
Classifier data loaded from default uri file:///D:/soft/audiveris-github/development/res/basic-clas
Versions. Poll frequency: Weekly, next poll on: 17-Jul-2021
```

Some quick explanations about these early messages :

1. Program version (e.g. 5.2.2)

2 Logging:

- Logging can be customized, here we are using a user-defined configuration file.
- All session messages are also written into a dedicated user file, named according to session date and time (e.g. 20210710T121739.log)

3 Configuration:

- User plugins if any to ease access to companion programs like MuseScore, Finale, etc.
- Glyph classifier trained data.

4 Check for new version on the GitHub site

Initial processing

1

Let’s load the input file O.lux.beata.Trinitas.Alberti.Johann.Friedrich.pdf into Audiveris 5.2.
A quick look at the images shows it is of rather good quality.
So, via `Book → Transcribe book` menu item, let’s launch the whole book transcription…
Done in 2 min 30 sec, about 30 sec per sheet.

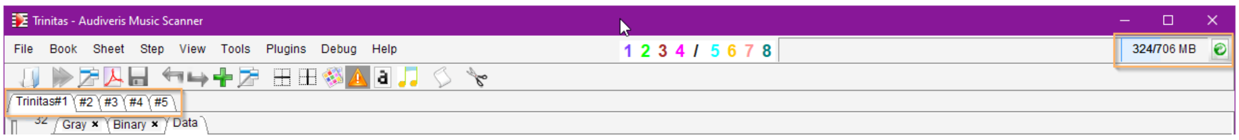
2

The last message says `[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Scores built: 3` which means that 3 movements have been transcribed in this book.

3 Before getting into user actions, let’s save the current project status, via the command `Book → Save Book` or the command `Book → Save Book as...`.
We choose the latter to shrink the project name from “O.lux.beata.Trinitas.Alberti.Johann.Friedrich” to simply “Trinitas”.

```
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /book.xml
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#1/BINARY.png
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#1/sheet#1.xml
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#2/BINARY.png
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#2/sheet#2.xml
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#3/BINARY.png
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#3/sheet#3.xml
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#4/BINARY.png
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#4/sheet#4.xml
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#5/BINARY.png
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Stored /sheet#5/sheet#5.xml
[O.lux.beata.Trinitas.Alberti.Johann.Friedrich] Book stored as D:\soft\cases\Issue-481\Trinitas.
```

4 Top of the Audiveris window is now like this:


Trinitas - Audiveris Music Scanner
File Book Sheet Step View Tools Plugins Debug Help
1 2 3 4 / 5 6 7 8
324/706 MB
Trinitas#1 #2 #3 #4 #5
Gray Binary Data

Notice the “memory meter” on the window upper right corner which gives 324/706 MB.

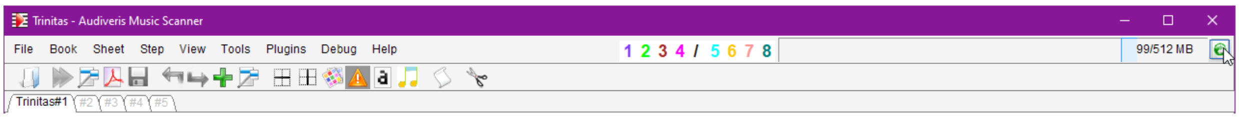
Notice also that we have the 5 sheets of the Trinitas book in memory. This is indicated by the 5 tabs on the window left side: there are 5 tabs because the book contains 5 sheets, and these tabs appear in standard font because they are currently loaded.

Five sheets is not a really high number, but let’s do something to save on memory, an action specifically useful for books with dozens of sheets or more:

The command `Book → Swap book sheets` gets rid of all sheets from memory – except the current sheet – after storing them to disk if needed.

```
Disposed sheet#2
Disposed sheet#3
Disposed sheet#4
Disposed sheet#5
```

The Audiveris window has changed:


Trinitas - Audiveris Music Scanner
File Book Sheet Step View Tools Plugins Debug Help
1 2 3 4 / 5 6 7 8
99/512 MB
Trinitas#1 #2 #3 #4 #5

All tabs, except tab #1, now appear in gray and the memory meter now shows 99/512 MB. We can select again any of these gray tabs, and the corresponding sheet gets reloaded from disk.

```
[Trinitas#2] Loaded /sheet#2/sheet#2.xml
[Trinitas#3] Loaded /sheet#3/sheet#3.xml
[Trinitas#4] Loaded /sheet#4/sheet#4.xml
[Trinitas#5] Loaded /sheet#5/sheet#5.xml
```

If we look carefully, we can see that, even with all sheets back in memory, the total memory consumption is just 164/531 MB, far below the initial 324/706. This is so because a bunch of transient data, needed to transcribe sheets, is no longer needed and has been disposed of, via the store/reload process.

Even more efficiently, we could also:

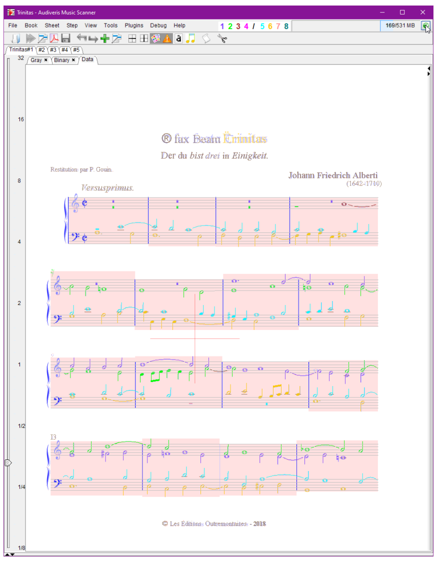
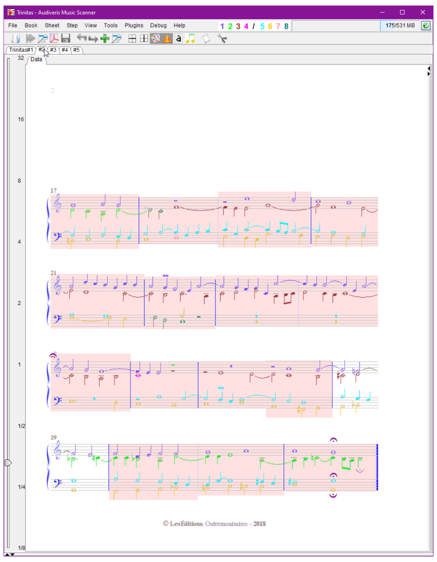
a Close the book using the `Book → Close book` command (or `Ctrl+W` shortcut). We would be prompted to save the book if needed.

b. Reopen the latest closed book (at its latest opened sheet), via the `Book → Most recent book` command (or `Ctrl+Shift+T` shortcut).
c. Going through all book sheets, the total memory would stay at about 105 MB.

Raw results

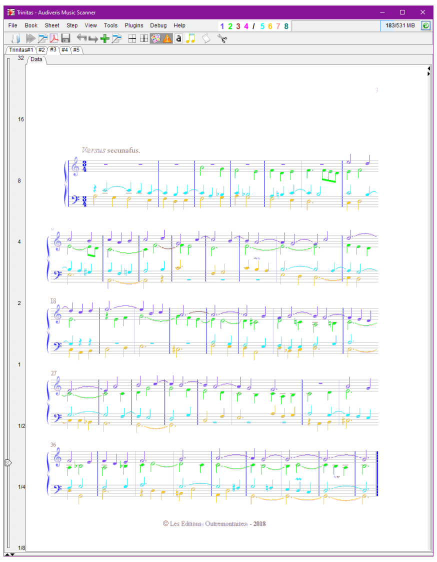
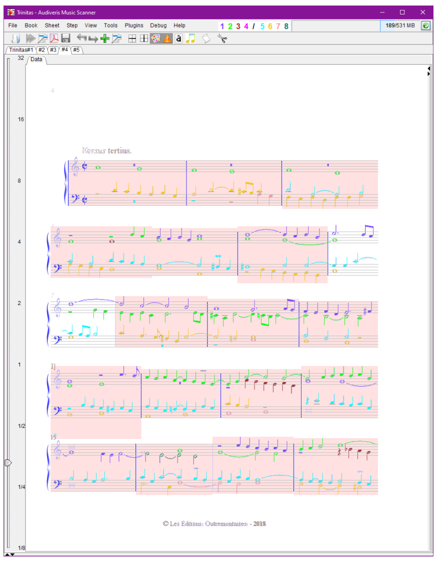
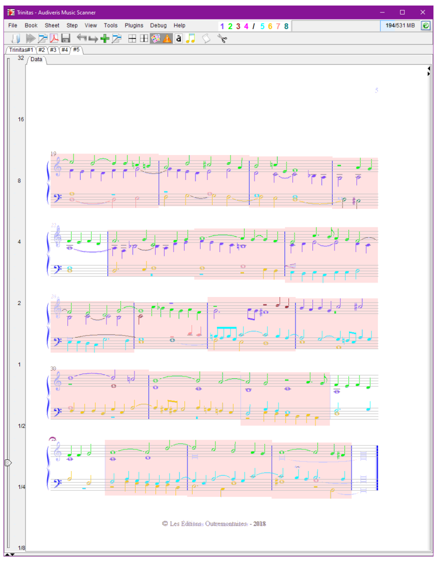
Here are the sheet results, right out of the OMR engine:
We have selected `View → Show score voices` or `F8` command so that voices in the same part are displayed in different colors.

> **NOTE**
> If you are reading this handbook via a web browser, you can ask the browser to display each of these sheet images in a separate tab.

Sheet ID	Raw result
Sheet#1	
Sheet#2	

Trinitas - Audiveris Music Scanner
File Book Sheet Step View Tools Plugins Debug Help
1 2 3 4 / 5 6 7 8
169/531 MB
Trinitas#1 #2 #3 #4 #5
Gray Binary Data
32
16
8
4
2
1
1/2
1/4
1/8
® fax Beam Trinitas
Der du bist drei in Einigkeit.
Restitution par P. Gouin.
Johann Friedrich Alberti
(1642-1710)
Versusprimus.
© Les Éditions Outremontaines - 2018

Trinitas - Audiveris Music Scanner
File Book Sheet Step View Tools Plugins Debug Help
1 2 3 4 / 5 6 7 8
175/531 MB
Trinitas#1 #2 #3 #4 #5
Data
32
16
8
4
2
1
1/2
1/4
1/8
© Les Éditions Outremontaines - 2018

Sheet ID	Raw result
Sheet#3	
Sheet#4	
Sheet#5	

Our first impression is that almost all measures, except for sheet #3, are displayed in pink!

A **pink** measure indicates that the **rhythm** analysis has failed for that measure:

- Either because some chord starting time could not be determined,
- Or because some voice was computed to complete after the theoretical measure end, as derived from the time signature.

So, what caused such a disastrous rhythm result on a rather good input?

We can double-check the time signatures. All have been correctly recognized:

- **Common-cut** at the beginning of sheet #1
- **3/4** at the beginning of sheet #3 (the one with no signaled errors)
- **Common-cut** at the beginning of sheet #4

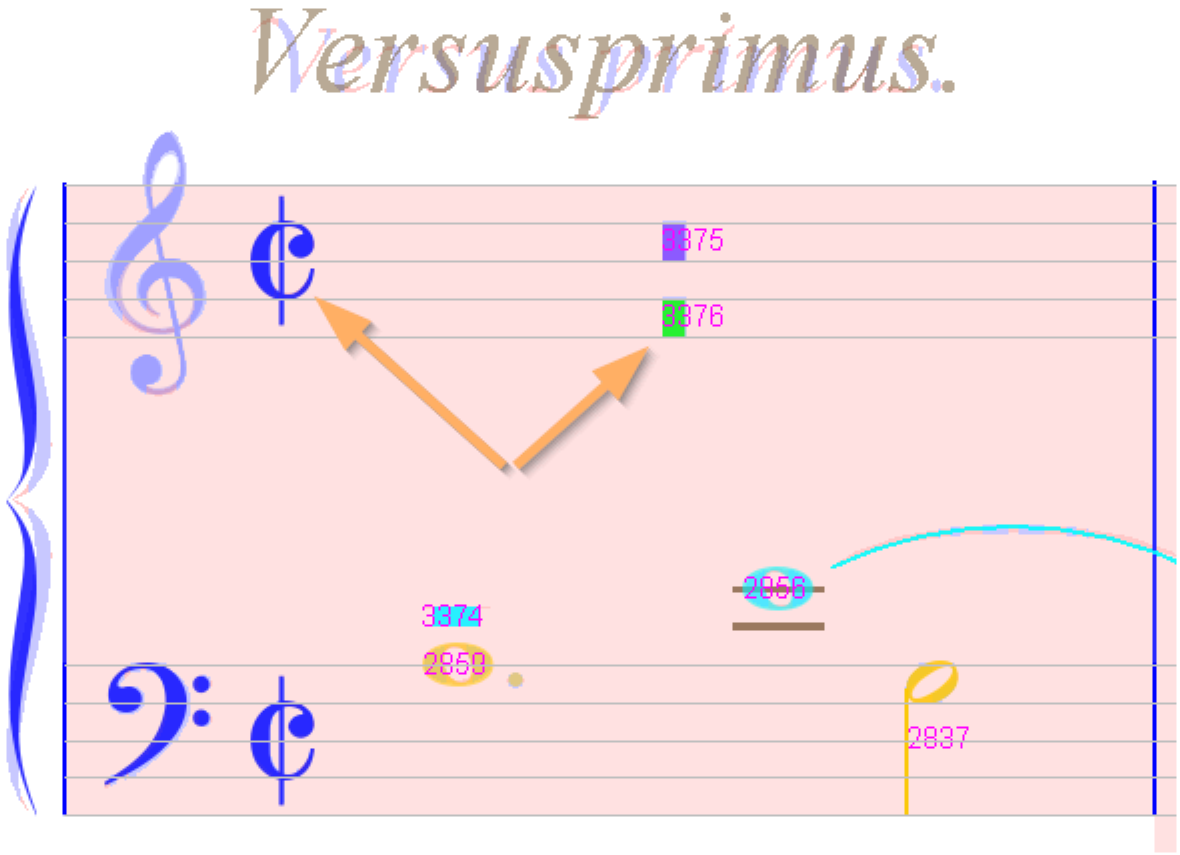
Let’s have a closer look, with colorized voices (`View → Show score voices`) and with chord IDs displayed (`View → Show chord IDs`).

> **NOTE**
> Chord IDs are dynamically assigned within each sheet by the Audiveris program. If we replay this processing even with just a slight difference in the engine code or parameters, some chord IDs may be different that those listed in this handbook version. The purpose of using chord IDs in this tutorial is to ease user mapping between snapshots and text lines.

Sheet #1

Sheet #1, Measure #1

The first measure of sheet #1 looks like:



1 Note the presence of “breve rests” (chords #3375 & #3376) on the upper staff. Such a multiple rest is supposed to last two bars.

2 Note also the presence in the lower staff of a dotted whole head (chord #2859), followed by a half note (chord #2837), for a total duration of 2 for voice #6.

So, there is a conflict between the actual duration of this measure (and of almost all other measures in the same page) with the time signature: 2 for each measure vs. 2/2=1 expected by the common-cut time signature.

What is the real “value” of a common-cut time signature? This Wikipedia Alla breve article sheds an interesting light, especially with the (1642-1710) date indication on the score top right corner. See also this Ultimate music theory article. Instead of 2/2, it would be worth 4/2.

So let’s replace all the common-cuts by explicit “4/2” time signatures.

To delete:

- We select the common-cut `Inter` and press `DEL` key.
- Or we click on the `Deassign` button in the `Inter` board.
- Or we use the `≡ Inters` contextual menu.

Inserting a 4/2 signature is more complex, since this is not one of the predefined time signatures:

1. From the Shape palette, we select the “Times” family (the family figured by a 4/4 sign) and from this shape family, we drag & drop the custom (0/0) time signature into the correct location.
2. Then in the `Inter` board, we replace the custom 0/0 value by 4/2 value and press `Enter`.

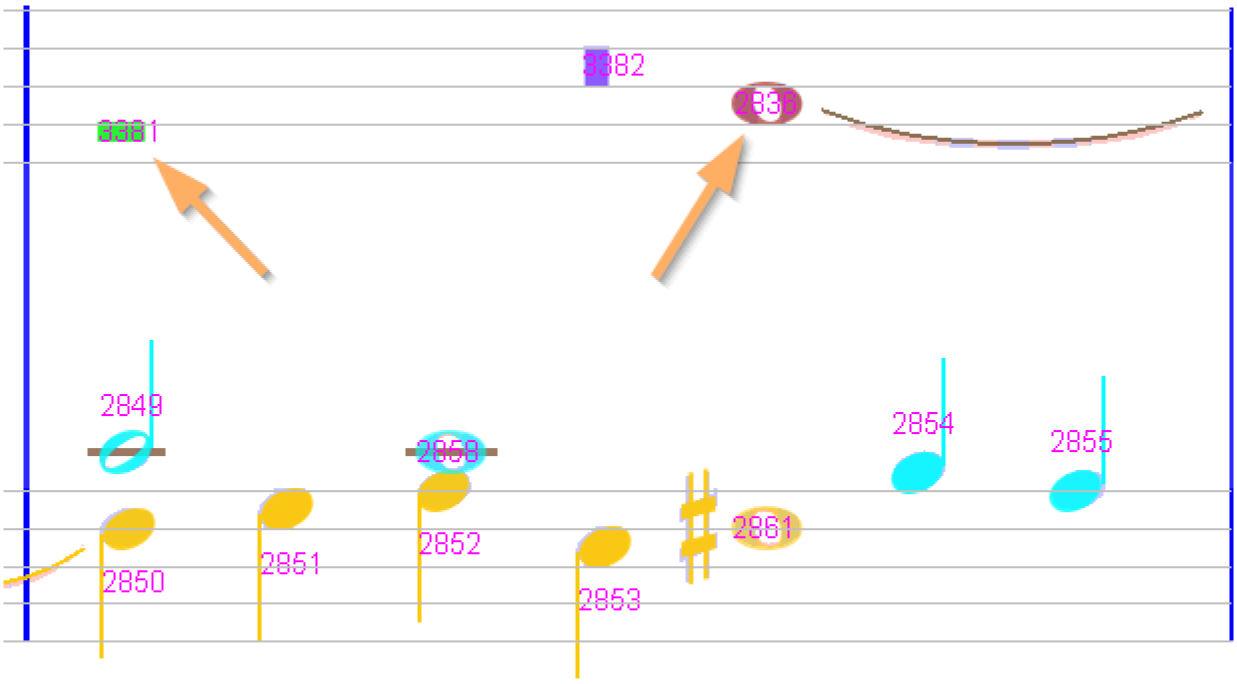
We do this for both common-cut signatures in the first measure of sheet #1. We’ll do the same thing on sheet #4 later.

> **NOTE**
> If we delete both original common-cut signatures before entering the new 4/2 values, we will notice that all measures of sheet #1, except measure #12, will turn white.
> The reason is that at this moment we have no time signature at all, hence no length-check can be performed on measures.
> [In measure #12 remaining pink, we’ll later see that a whole note has not been recognized].

With 4/2 clearly stated, among the 16 measures in the sheet, we now have 11 in white. A significant improvement! :-)

Sheet #1, Measure #4

All 4 measures of the first system look nice, with 2 voices in the upper staff and 2 other voices in the lower staff, except the last one (measure #4) which exhibits 3 voices in the upper staff:

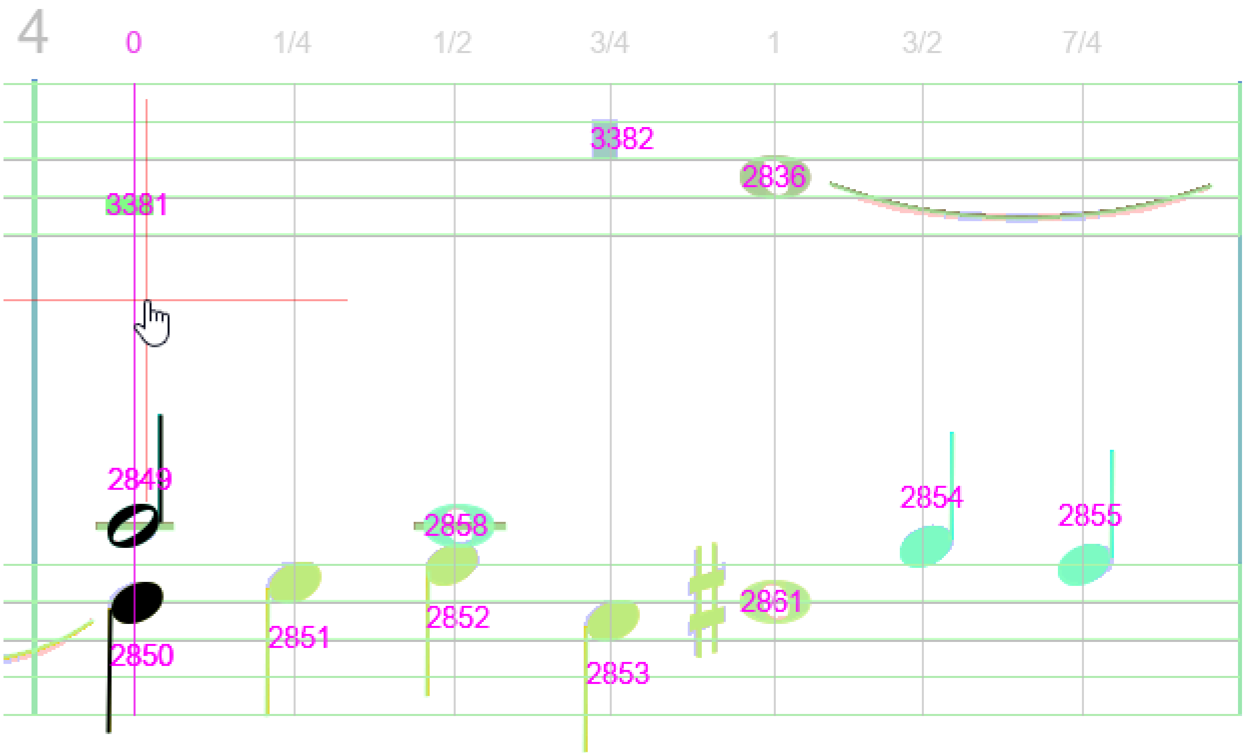


Referring to voice legend 1 2 3 4 / 5 6 7 8, this upper staff contains:

1. Voice #1, chord #3382 made of a breve rest, horizontally located at the measure center,
2. Voice #2, chord #3381 made of a whole rest, located at the beginning of the measure,
3. Voice #3, chord #2836 made of a whole note, located at 2/3 of the measure.

The problem is with the whole rest (chord #3381) which, as a measure-long rest, is the only chord in its voice but should be located at measure center. In fact, and there are other similar cases in this book, the whole rest is never at measure center but located as a plain chord in a time slot.

We can visualize the content of the first slot, using the combined mode (♫) and pressing the right mouse button near the first slot. Chord #3381 is *not* part of this slot:



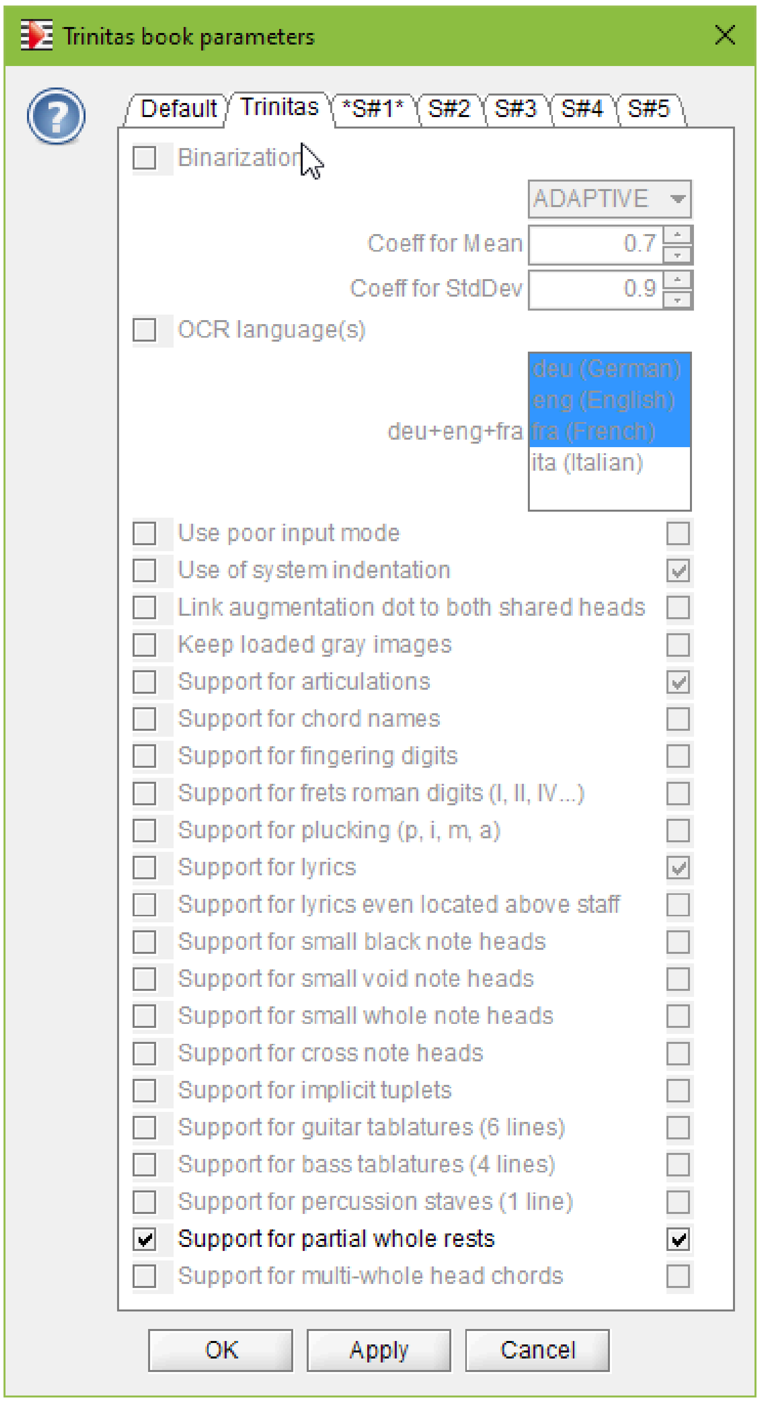
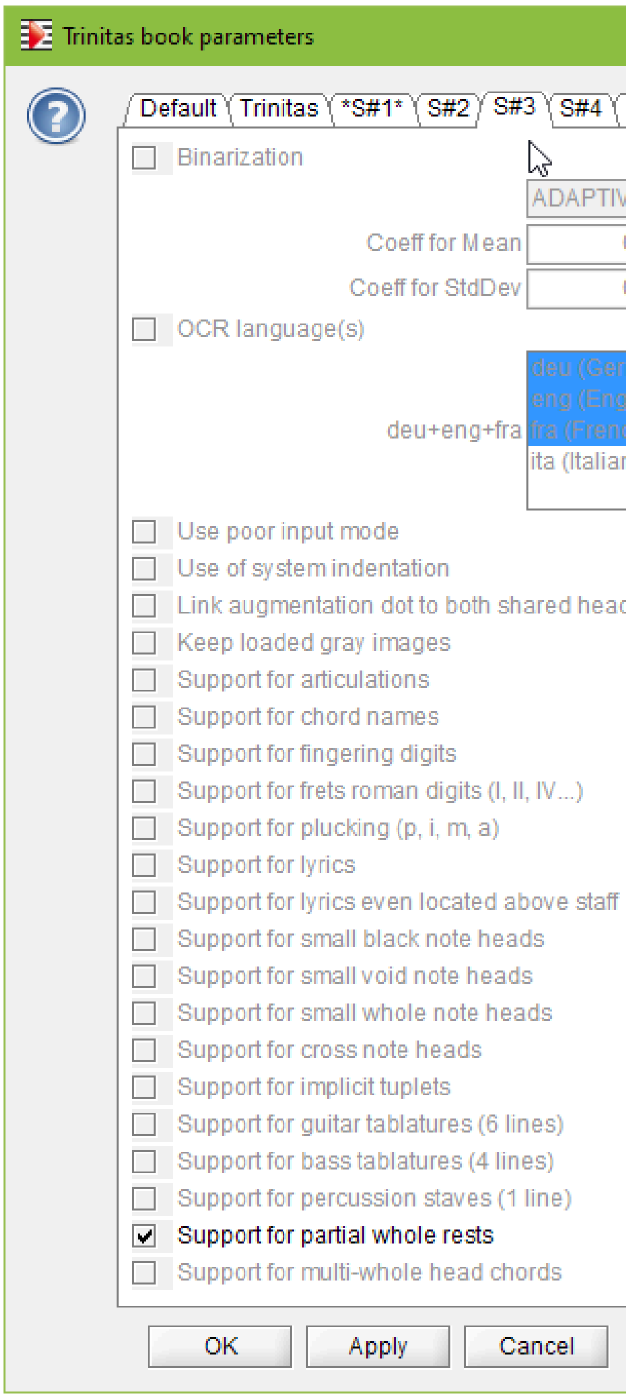
This is the purpose of a new option accessible via `Book → Set book parameters`, and named “Support for partial whole rests”. When this option is on for the sheet at hand, whole rests are no longer considered as measure-long rests, they are just plain rests with a duration of 1/1:

- They belong to a time slot,
- Their voice can contain other notes or rests.

So, we open the `Book → Set book parameters` dialog, and:

1 We *set* this option for the whole book (so that it applies to all sheets in book)

2 We explicitly *unset* this option for sheet #3.

Whole Trinitas Book	Except Trinitas Sheet #3
	

Trinitas book parameters

Default | Trinitas | *S#1* | S#2 | S#3 | S#4 | S#5

☐ Binarization

ADAPTIVE

Coeff for Mean 0.7

Coeff for StdDev 0.9

☐ OCR language(s)

deu+eng+fra

deu (German)
eng (English)
fra (French)
ita (Italian)

☐	Use poor input mode	☐
☐	Use of system indentation	☑
☐	Link augmentation dot to both shared heads	☐
☐	Keep loaded gray images	☐
☐	Support for articulations	☑
☐	Support for chord names	☐
☐	Support for fingering digits	☐
☐	Support for frets roman digits (I, II, IV...)	☐
☐	Support for plucking (p, i, m, a)	☐
☐	Support for lyrics	☑
☐	Support for lyrics even located above staff	☐
☐	Support for small black note heads	☐
☐	Support for small void note heads	☐
☐	Support for small whole note heads	☐
☐	Support for cross note heads	☐
☐	Support for implicit tuplets	☐
☐	Support for guitar tablatures (6 lines)	☐
☐	Support for bass tablatures (4 lines)	☐
☐	Support for percussion staves (1 line)	☐
☑	Support for partial whole rests	☑
☐	Support for multi-whole head chords	☐

OK | Apply | Cancel

Trinitas book parameters

Default | Trinitas | *S#1* | S#2 | S#3 | S#4

☐ Binarization

ADAPTIV

Coeff for Mean

Coeff for StdDev

☐ OCR language(s)

deu+eng+fra

deu (Ger
eng (Eng
fra (Frenc
ita (Italiar

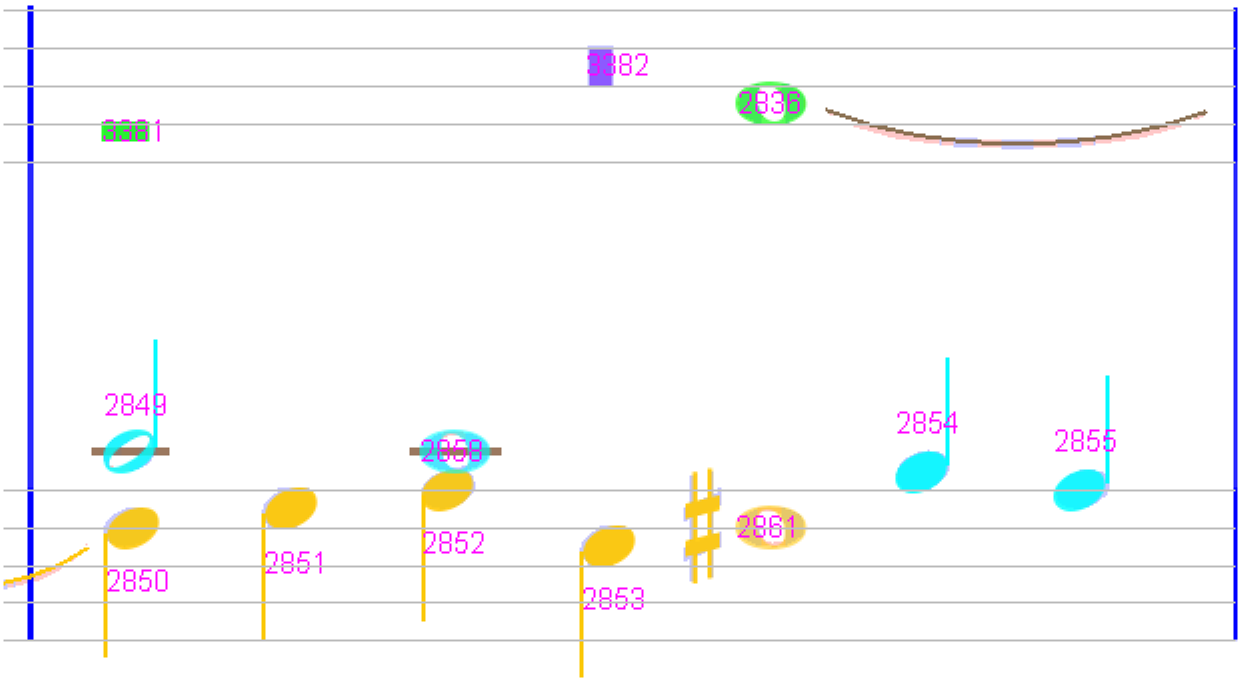
☐	Use poor input mode
☐	Use of system indentation
☐	Link augmentation dot to both shared heac
☐	Keep loaded gray images
☐	Support for articulations
☐	Support for chord names
☐	Support for fingering digits
☐	Support for frets roman digits (I, II, IV...)
☐	Support for plucking (p, i, m, a)
☐	Support for lyrics
☐	Support for lyrics even located above staff
☐	Support for small black note heads
☐	Support for small void note heads
☐	Support for small whole note heads
☐	Support for cross note heads
☐	Support for implicit tuplets
☐	Support for guitar tablatures (6 lines)
☐	Support for bass tablatures (4 lines)
☐	Support for percussion staves (1 line)
☑	Support for partial whole rests
☐	Support for multi-whole head chords

OK | Apply | Cancel

We don’t forget to press the `OK` or `Apply` button to commit the parameters.

Then, we use the contextual `≡ Page #1 → Reprocess rhythm` to update the page.

Measure #4 now looks like this (notice chords #3381 and #2836 are now in the same green voice):



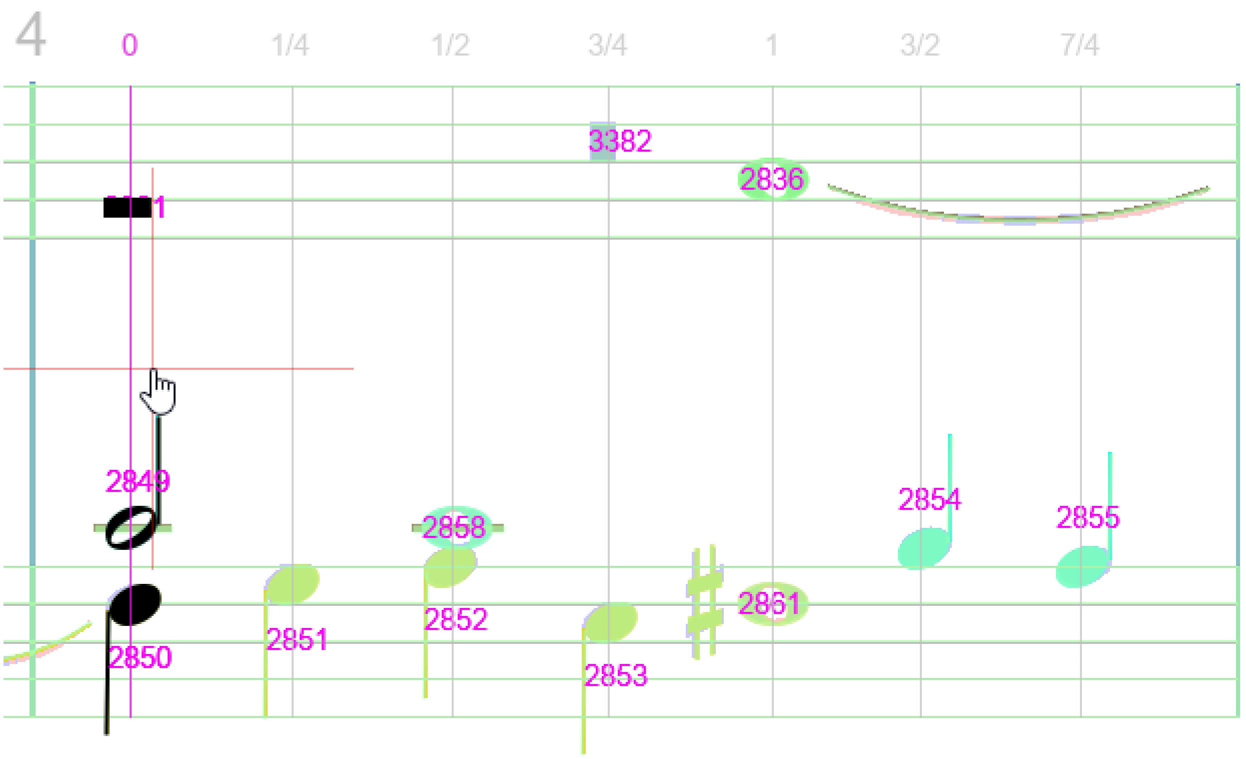
And `≡ Measure #4 → Dump stack voices` dumps its strip as:

```
MeasureStack#4
    |0       |1/4     |1/2     |3/4     |1       |3/2     |7/4     |2
--- P1
V 1 |Ch#3382 ======================================================|M
V 2 |Ch#3381 ===========================|Ch#2836 ==================|2 (ts:2/1)
V 5 |Ch#2849 =========|Ch#2858 ==================|Ch#2854 |Ch#2855 |2
V 6 |Ch#2850 |Ch#2851 |Ch#2852 |Ch#2853 |Ch#2861 ==================|2
```

We can see that voice #1 end time is displayed as ‘M’ (M for Measure), meaning full measure duration, which is OK for a measure-long rest such as the breve rest.

Also, voice #2 now contains chord #3381 (a whole rest, but not a measure-long rest) followed by chord #2836 (a whole note), to sum up to value 2, the same as the following voices in measure.

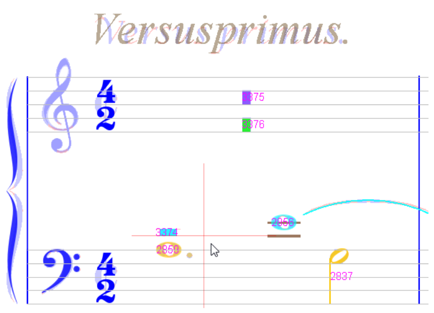
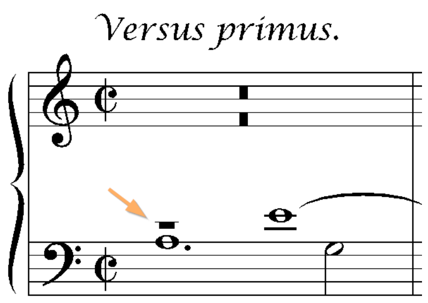
Here is the updated view of the first slot content (which now includes the whole rest in the upper staff):



Sheet #1, Measure #1 (again)

This “Support for partial whole rest” was key to fix rhythms of measures like measure #4 we just saw.

But, before we set this option, measure #1 already appeared OK. In fact, its whole rest of chord #3374 had been mistaken for a half rest:

Data view	Binary view
	

If we click on this rest, the `Inter` board gives HALF_REST shape. But the binary view clearly indicates it should be a WHOLE_REST shape.

The measure strip confirms the fact that chord #3374 duration is only 1/2, resulting in a voice ending at 3/2 instead of 2:

```
MeasureStack#1
    |0       |1/2     |3/2     |2
--- P1
```

```
V 1 |Ch#3375 ==================|M
V 2 |Ch#3376 ==================|M
V 5 |Ch#3374 |Ch#2856 |........|3/2
V 6 |Ch#2859 =========|Ch#2837 |2
```

In fact, this is a limitation of the current engine with its glyph classifier. Within a staff height, with the staff lines “removed”, the glyph for a half rest and the glyph for a whole rest are nearly identical – a horizontal rectangle. Only the position relative to the staff line nearby allows the engine to tell the difference between:

- a whole rest – close to the staff line above –
- and a half rest – close to the staff line below –

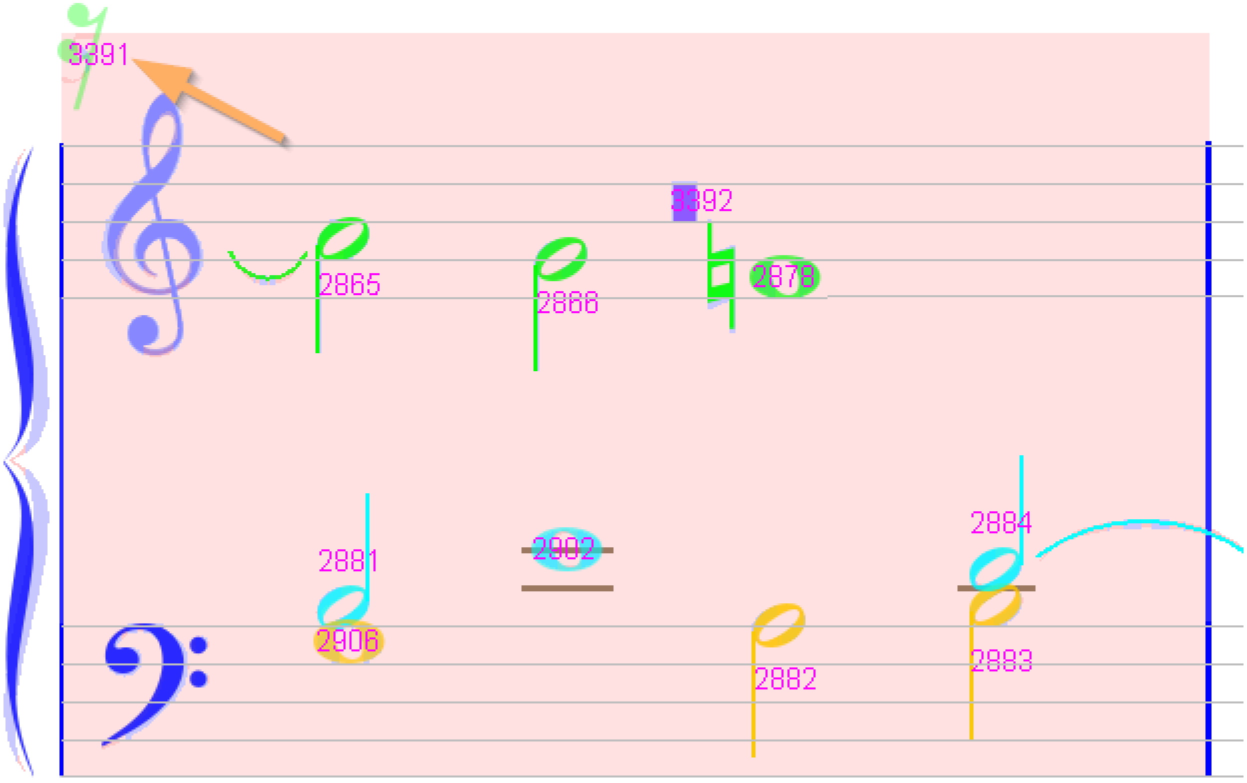
In the case at hand, the rest glyph is badly located with respect to the staff (see its base aligned with the next ledger on its right side), and it was thus mistaken for a half rest.

No big deal, we can easily delete this incorrect inter and, re-using its underlying glyph, manually create a true whole rest inter.

```
MeasureStack#1
    |0       |1       |3/2     |2
--- P1
V 1 |Ch#3375 ==================|M
V 2 |Ch#3376 ==================|M
V 5 |Ch#3420 |Ch#2856 =========|2 (ts:2/1)
V 6 |Ch#2859 =========|Ch#2837 |2
```

A new chord (#3420) has replaced the former chord #3374, and the measure is now OK.

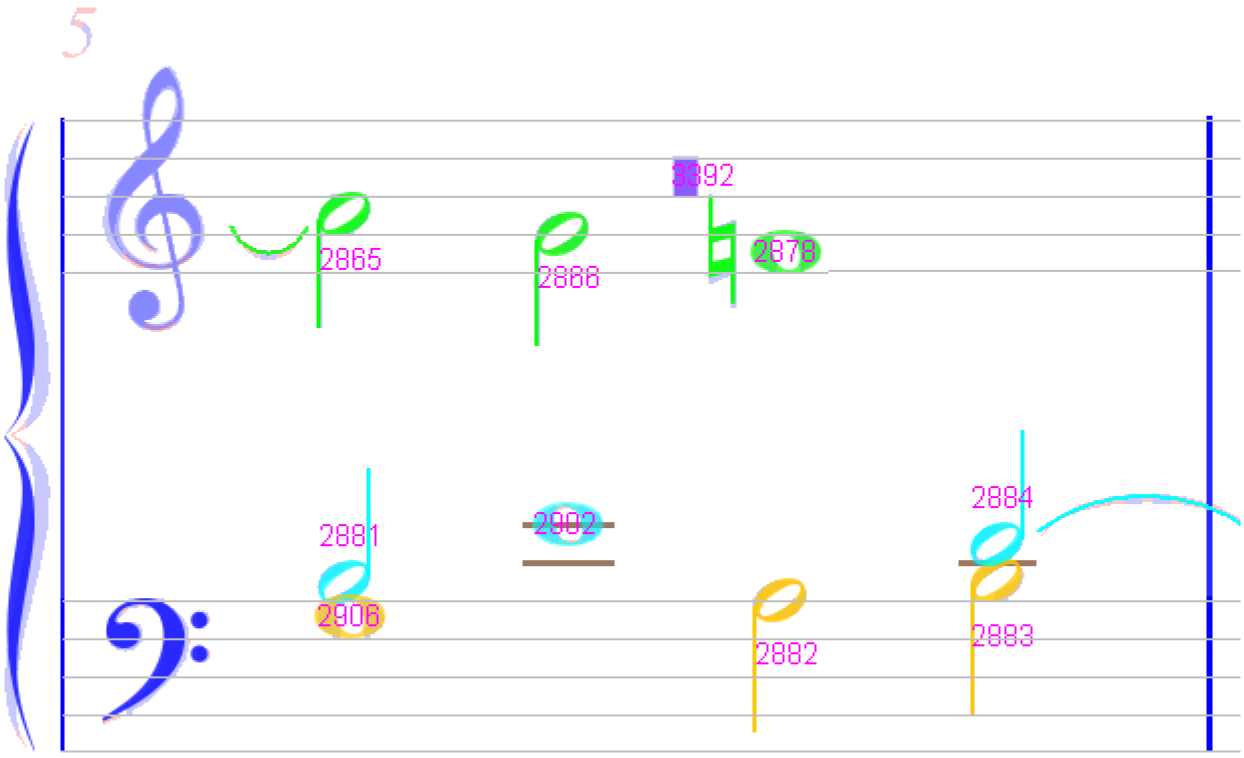
Sheet #1, Measure #5



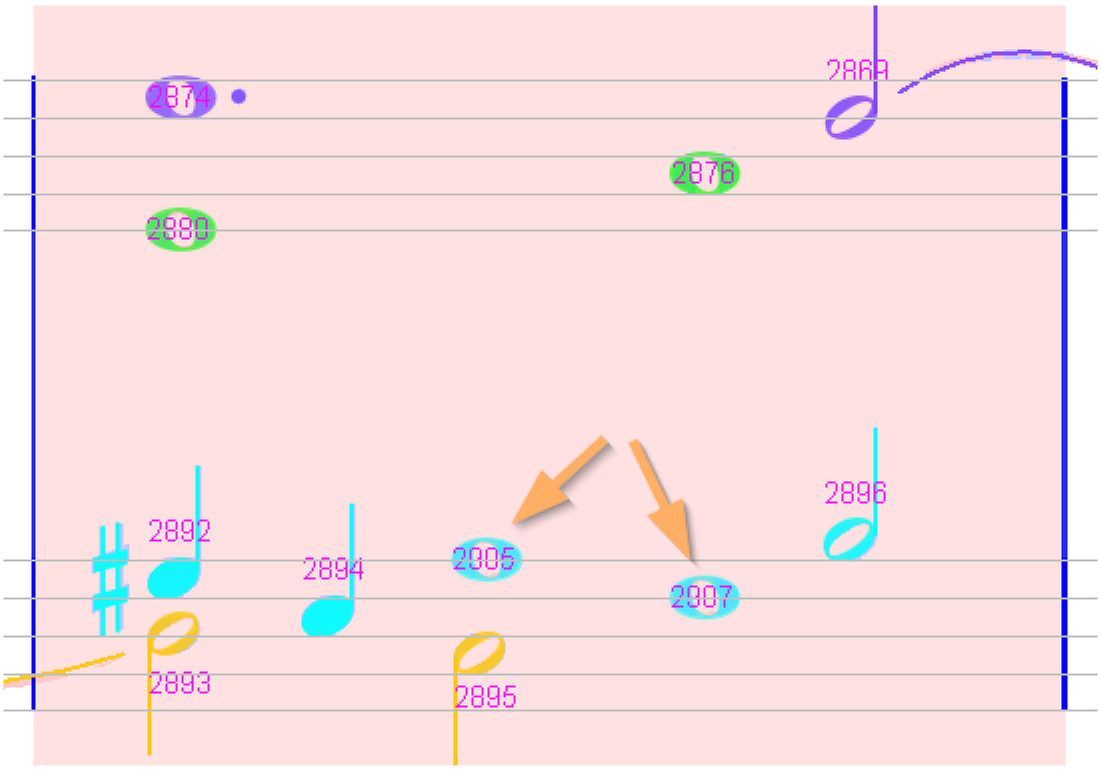
This pink measure exhibits a rather obvious problem:

- A false 16th rest in the upper left corner (chord #3391).
  Hint: the measure background goes much too high with respect to the staff line, indicating some abnormal chord located there. We can simply delete it.

This modification automatically triggers a rhythm update for the measure at hand:



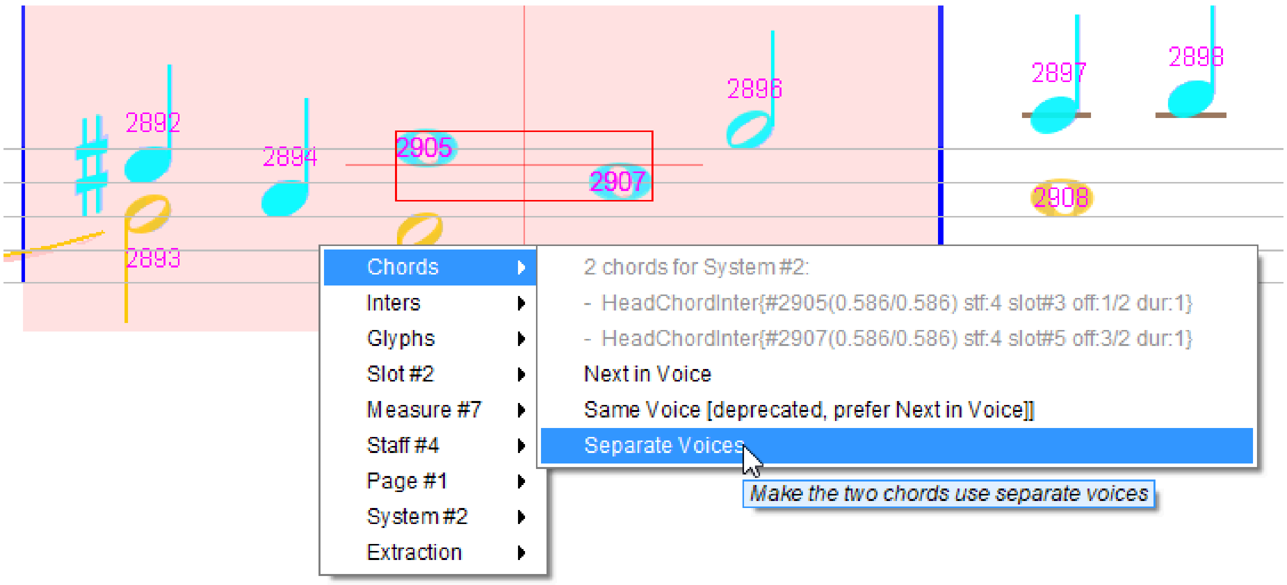
Sheet #1, Measure #7

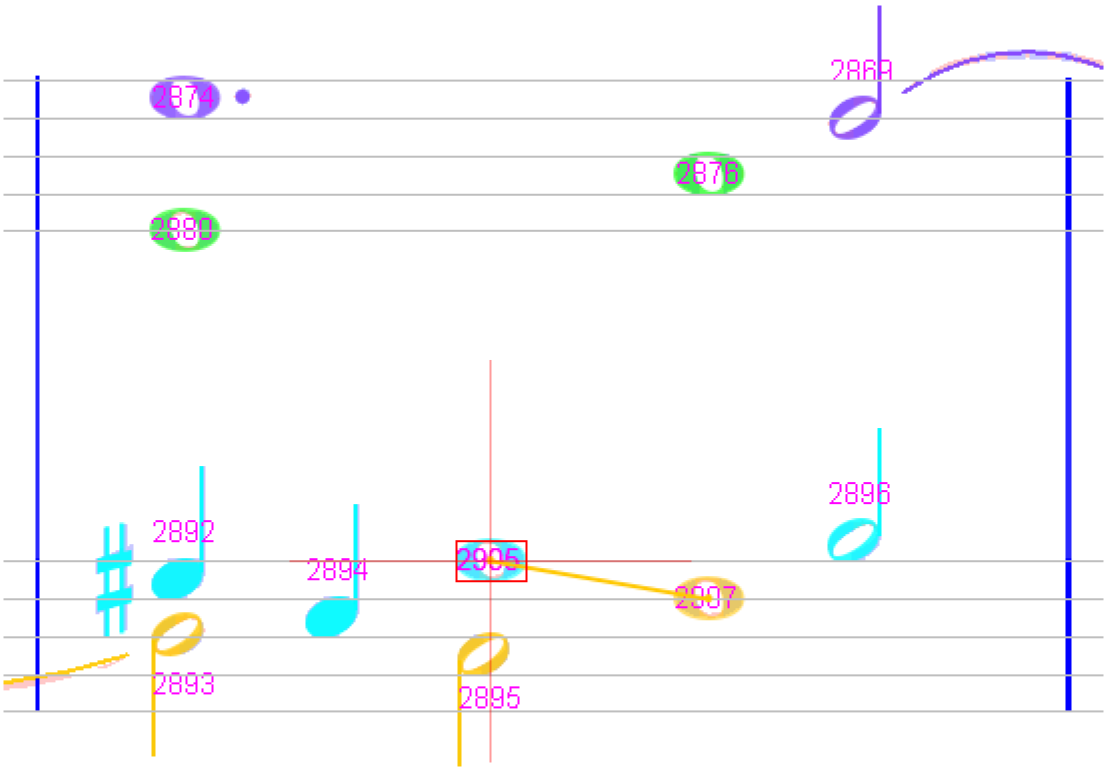


The lower staff of this pink measure exhibits a sequence of 2 whole notes, together with additional quarter and half notes, all in the same voice! This cannot fit within a 4/2 measure duration, and is confirmed by the measure strip:

```
MeasureStack#7
    |0       |1/4     |1/2     |1       |3/2     |5/2     |3
--- P1
V 1 |Ch#2874 ===========================|Ch#2869 |........|2
V 2 |Ch#2880 ==================|Ch#2876 =========|........|2 (ts:2/1)
V 5 |Ch#2892 |Ch#2894 |Ch#2905 =========|Ch#2907 |Ch#2896 |3
V 6 |Ch#2893 =========|Ch#2895 |........|........|........|1
```

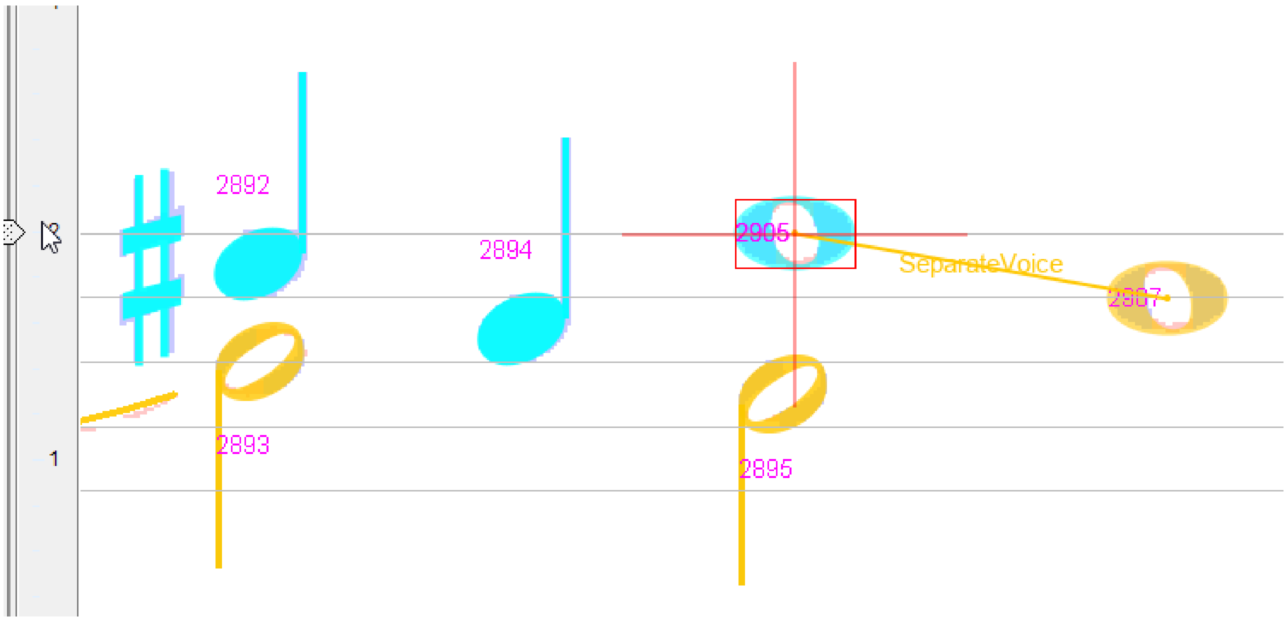
The fix is to force these 2 whole notes (chords #2905 & #2907) to be in separate voices:


2892
2894
2905
2907
2896
2897
2898
2908
2893
Chords
Inters
Glyphs
Slot #2
Measure #7
Staff #4
Page #1
System #2
Extraction
2 chords for System #2:
- HeadChordInter{#2905(0.586/0.586) stf:4 slot#3 off:1/2 dur:1}
- HeadChordInter{#2907(0.586/0.586) stf:4 slot#5 off:3/2 dur:1}
Next in Voice
Same Voice [deprecated, prefer Next in Voice]]
Separate Voices
Make the two chords use separate voices

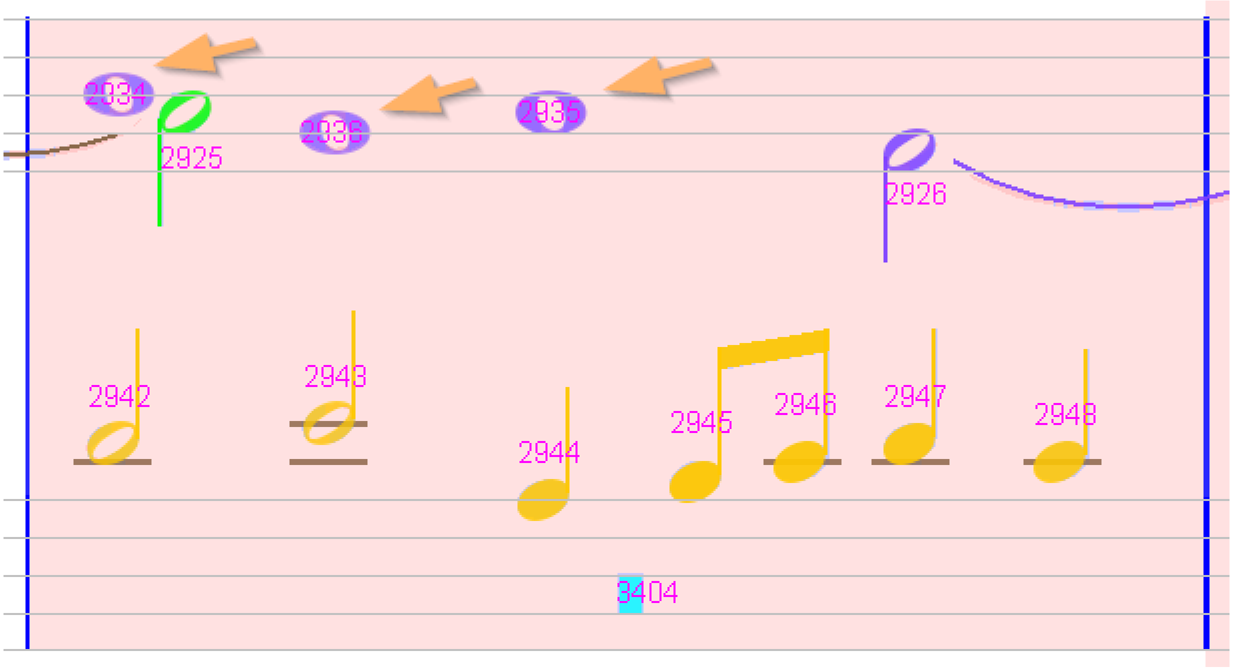


Since chord #2905 is still selected, it displays the “Separate Voice” relation with chord #2907 as an orange-colored segment.

If we increase the zoom ratio to 2 or above, there is now enough display room, and the relation name “SeparateVoice” explicitly appears:



Sheet #1, Measure #11

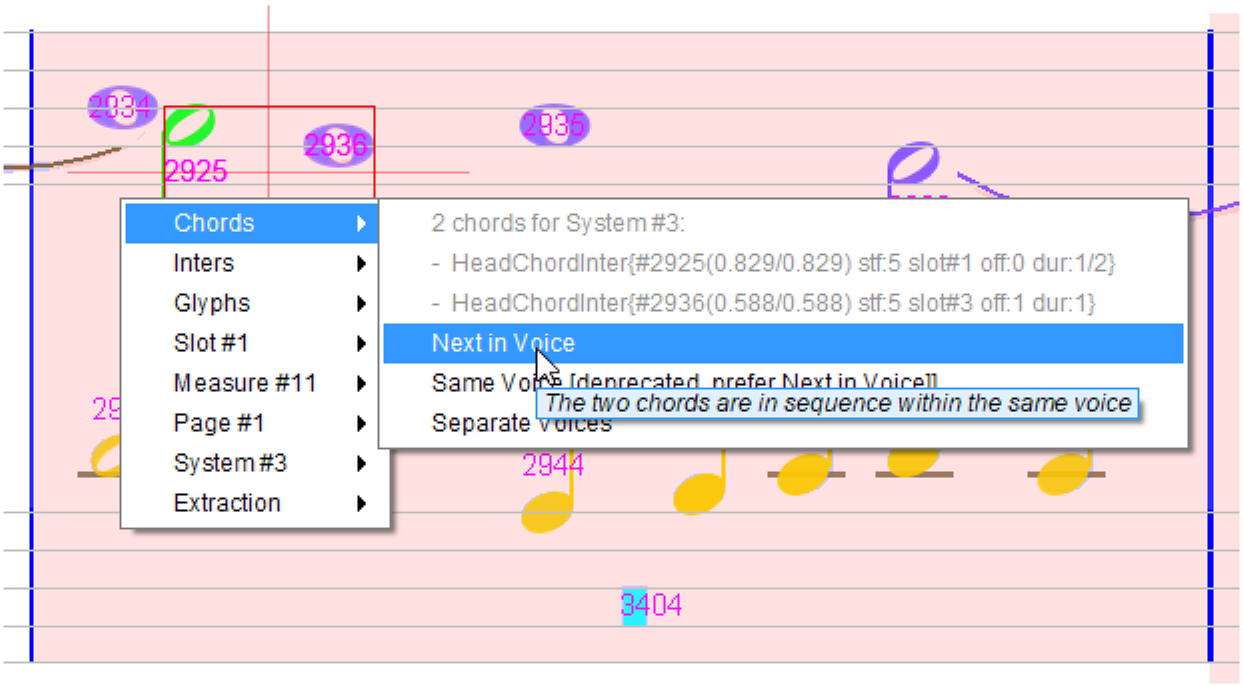


On the upper staff, 3 whote notes appear in the same voice #1 which is really too much, while a half note stays alone in voice #2!

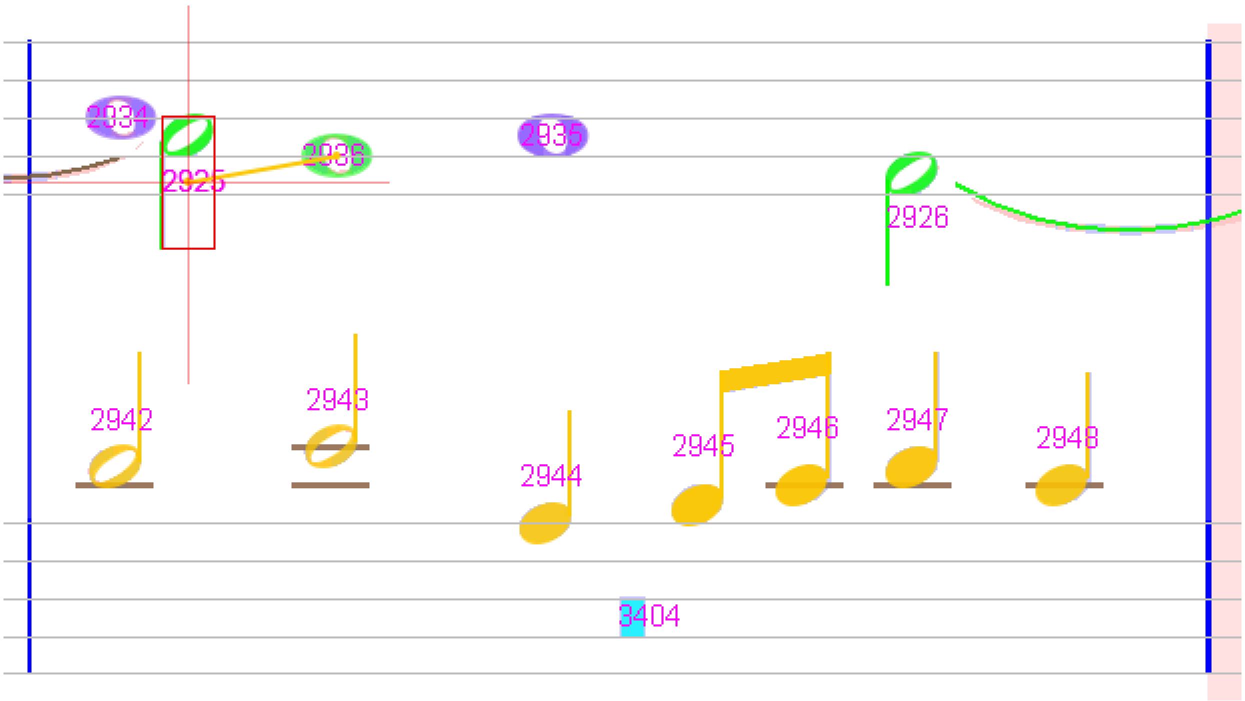
We can either separate the first 2 wholes (chords #2934 and #2936)
or we can declare the half note (chord #2925) and the whole chord #2936 as being in sequence in the same voice.

Either way is OK.

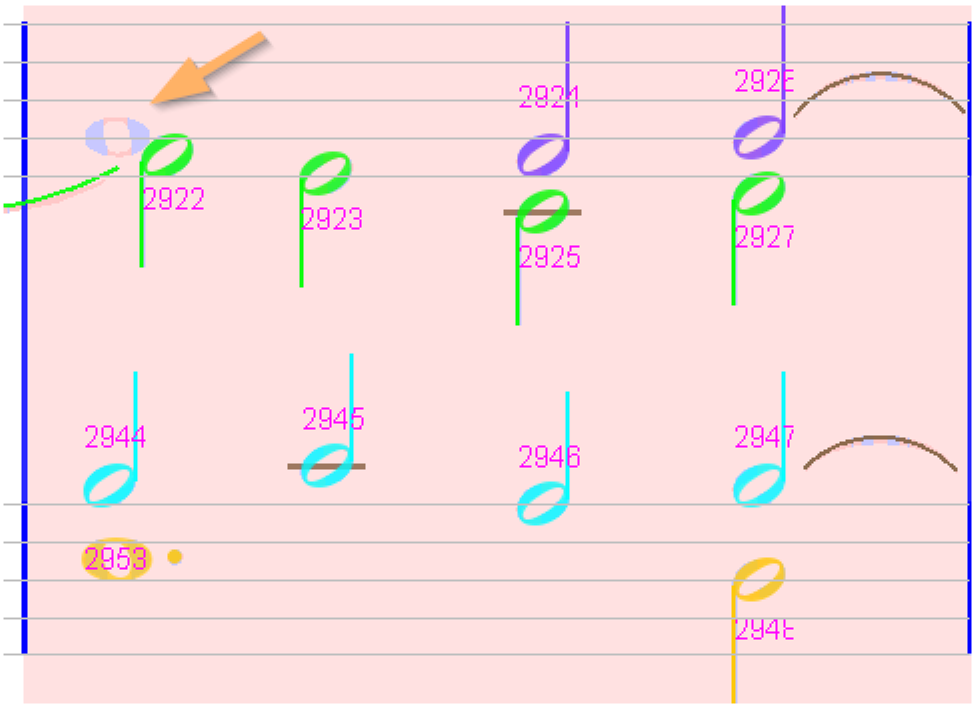
To choose the latter, we select the “Next In Voice” relation between #2925 and #2936.


2934
2925
2936
2935
29
2944
3404
Chords
Inters
Glyphs
Slot #1
Measure #11
Page #1
System #3
Extraction
2 chords for System #3:
- HeadChordInter{#2925(0.829/0.829) stf:5 slot#1 off:0 dur:1/2}
- HeadChordInter{#2936(0.588/0.588) stf:5 slot#3 off:1 dur:1}
Next in Voice
Same Voice [deprecated, prefer Next in Voice]
The two chords are in sequence within the same voice
Separate Voices

And result is now OK:



Sheet #1, Measure #12



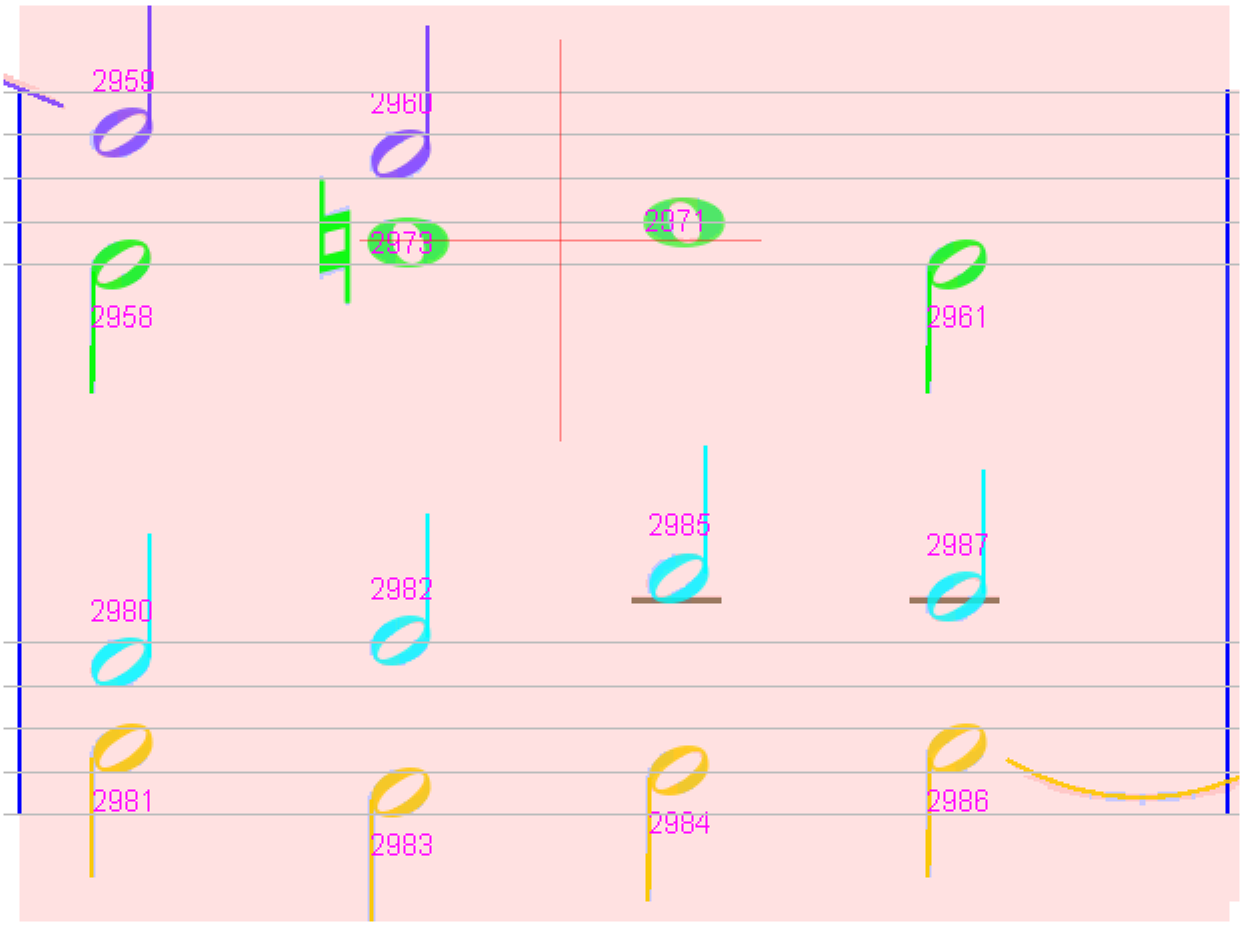
A missed whole note at the beginning of the upper staff.

We select the glyph (which is almost intact, despite the half head on the right) and assign it the whole note shape using the shape palette.

To do so, in the shape palette, we select the “HeadsAndDot” family figured by a black head and, while the target glyph is still selected, we use a double-click on the WHOLE_NOTE icon.

> **IMPORTANT**
> The OMR engine uses a specific technique (*template matching*) to recognize heads.
> The glyph classifier works correctly for isolated fixed-size symbols like a treble clef but would work very poorly on heads because their underlying glyphs are often difficult to retrieve (think of the glyph merge between head and staff lines and/or between head and stem).
> So, don’t even think of the Glyph Classifier top 5 output when dealing with heads!

Sheet #1, Measure #14



Yet another example of 2 whole notes (chord #2973 & #2971) with other notes in the same voice. The fix is to force separate voices.

In this rather simple case, we could improve the OMR engine algorithm by checking the current voice end time when deciding to append or not the second whole note to voice #2. Room for improvement!

This concludes our manual work on Sheet #1, with no rhythm errors left.

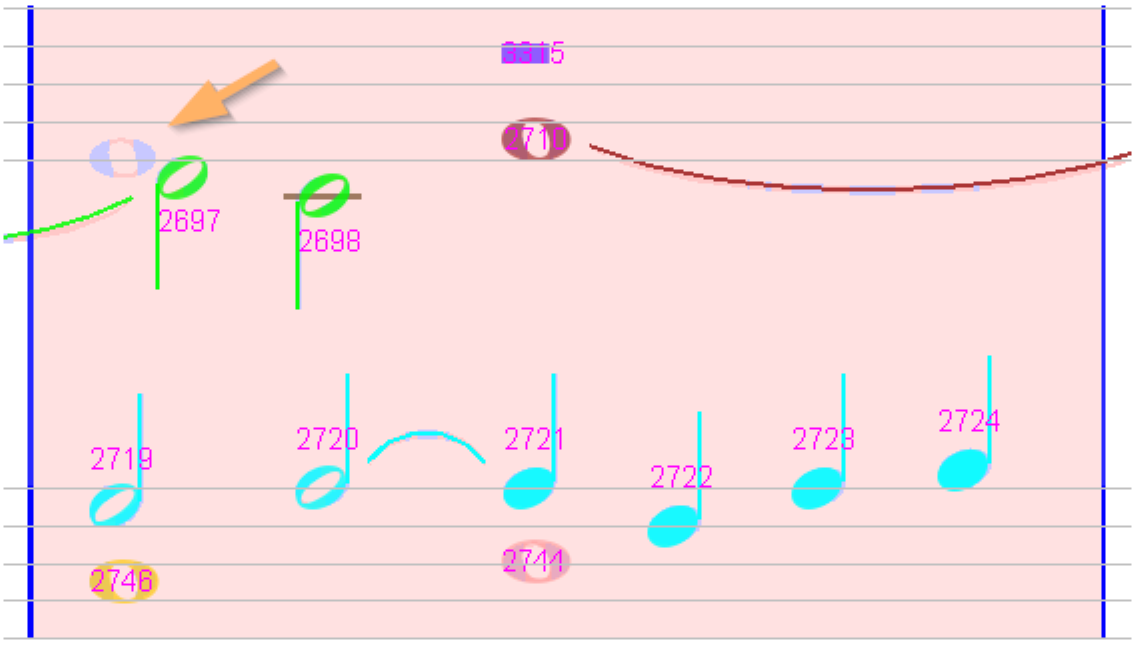
Sheet #2

We update the rhythm on the whole sheet #2, via `≡ Page #1 → Reprocess rhythm`, to benefit from the time signature fix (4/2) made on sheet #1.

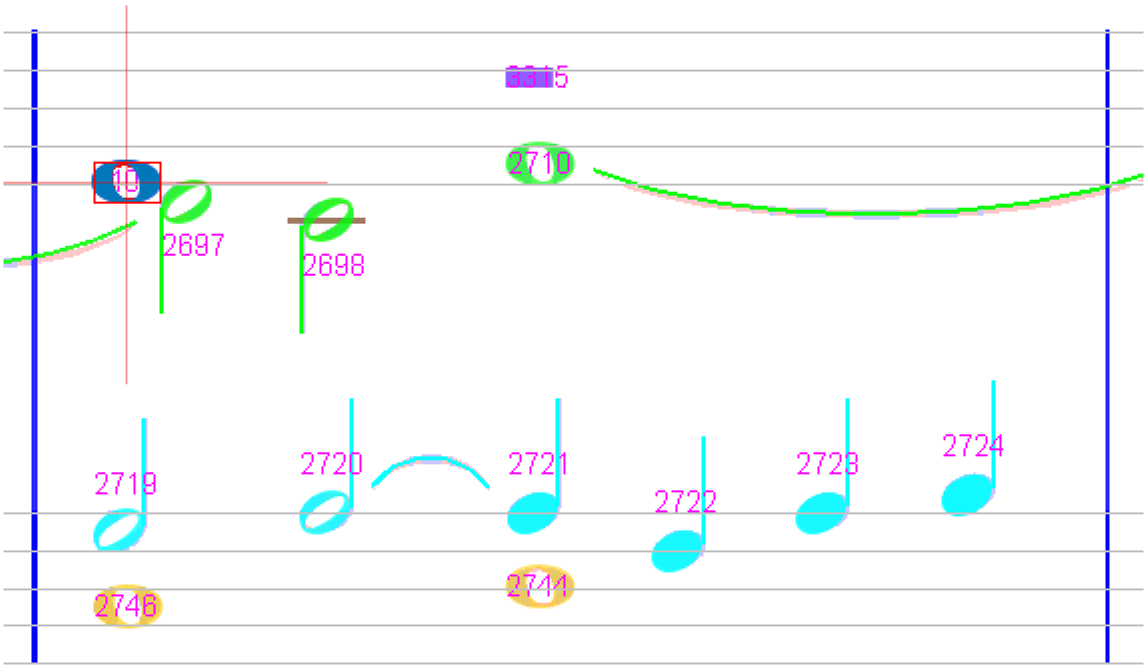
Only 3 measures are left in pink, but let’s review all measures.

> **NOTE**
> While we work an a sheet, measure numbers are local to the sheet (to the page to be precise). Hence, even though sheet #2 image starts with a number “17”, local measure numbers start at 1.
> Don’t worry, measure numbers will be consolidated when scores are refined at the end of book transcription.

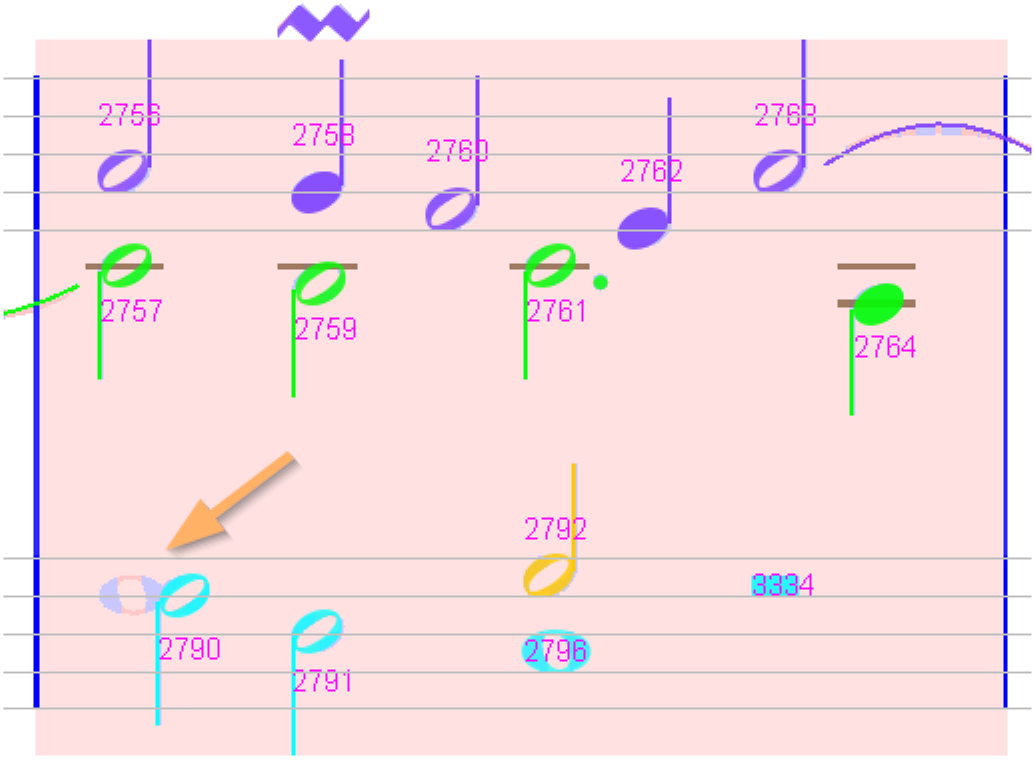
Sheet #2, Measure #2



Again a missed whole note at the upper left. Inserting it manually makes the entire measure OK, even for the voices of the lower staff.

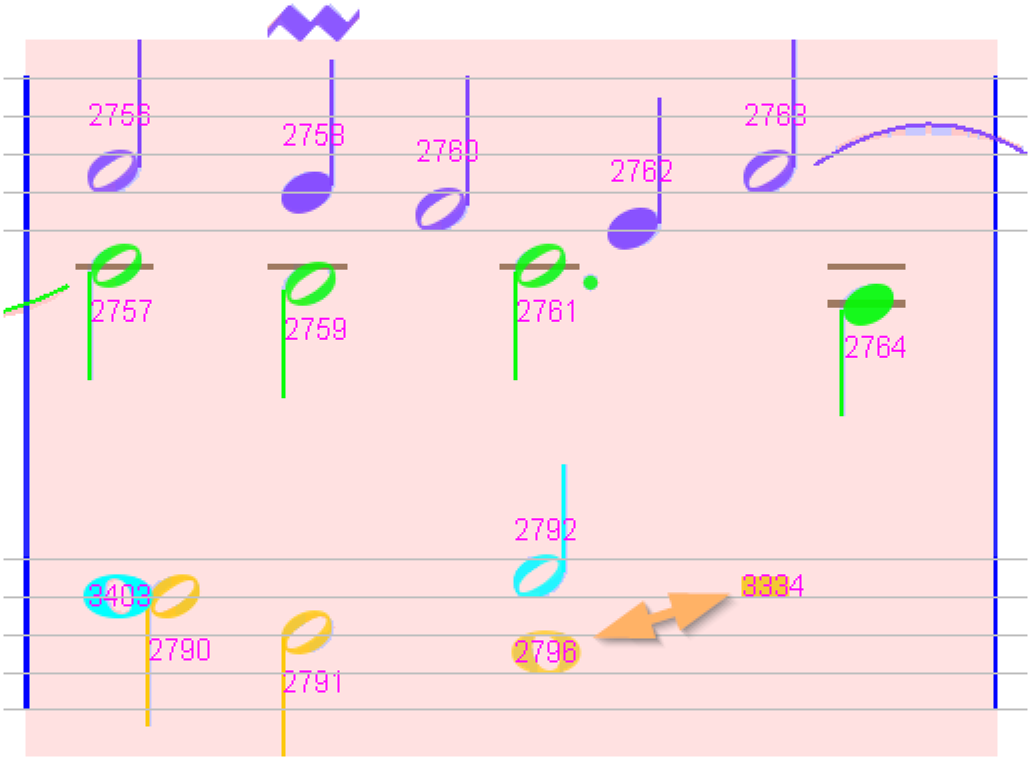


Sheet #2, Measure #6



A missed whole note at the beginning of the lower staff. We create it manually.

However, the measure remains in pink:



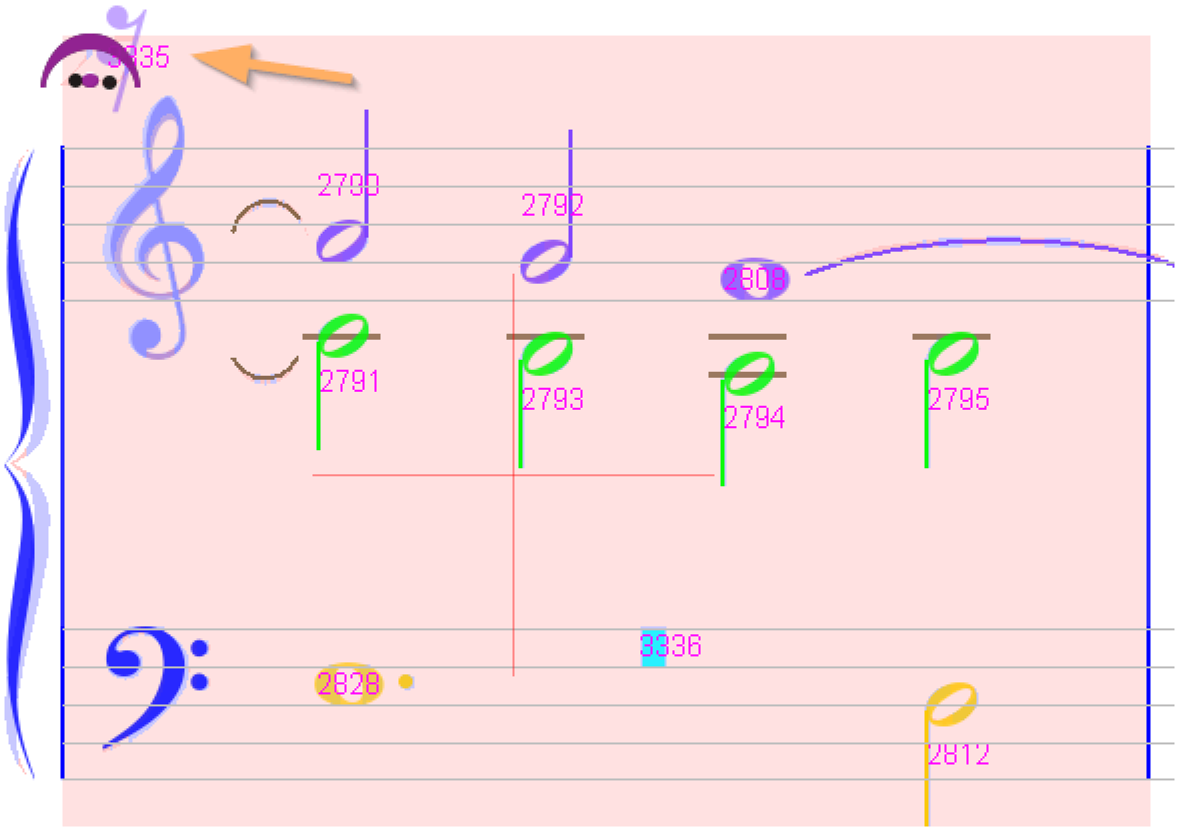
```
MeasureStack#6
    |0        |1/2      |3/4      |1        |5/4      |3/2      |7/4      |2        |5/2
--- P1
V 1 |Ch#2756  |Ch#2758  |Ch#2760  =========|Ch#2762  |Ch#2763  =========|........|2
```

```
V 2 |Ch#2757 |Ch#2759 =========|Ch#2761 ==================|Ch#2764 |........|2
V 5 |Ch#3403 ==================|Ch#2792 =========|........|........|........|3/2
V 6 |Ch#2790 |Ch#2791 =========|Ch#2796 ===========================|Ch#3334 |5/2
```

The half rest at the end of lower staff (chord #3334) should not be linked to whole chord #2796 but to half chord #2792.
So we force either “Separate Voices” between #2796 and #3334 or “Next in Voice” between #2792 and #3334 to get the correct rhythm:

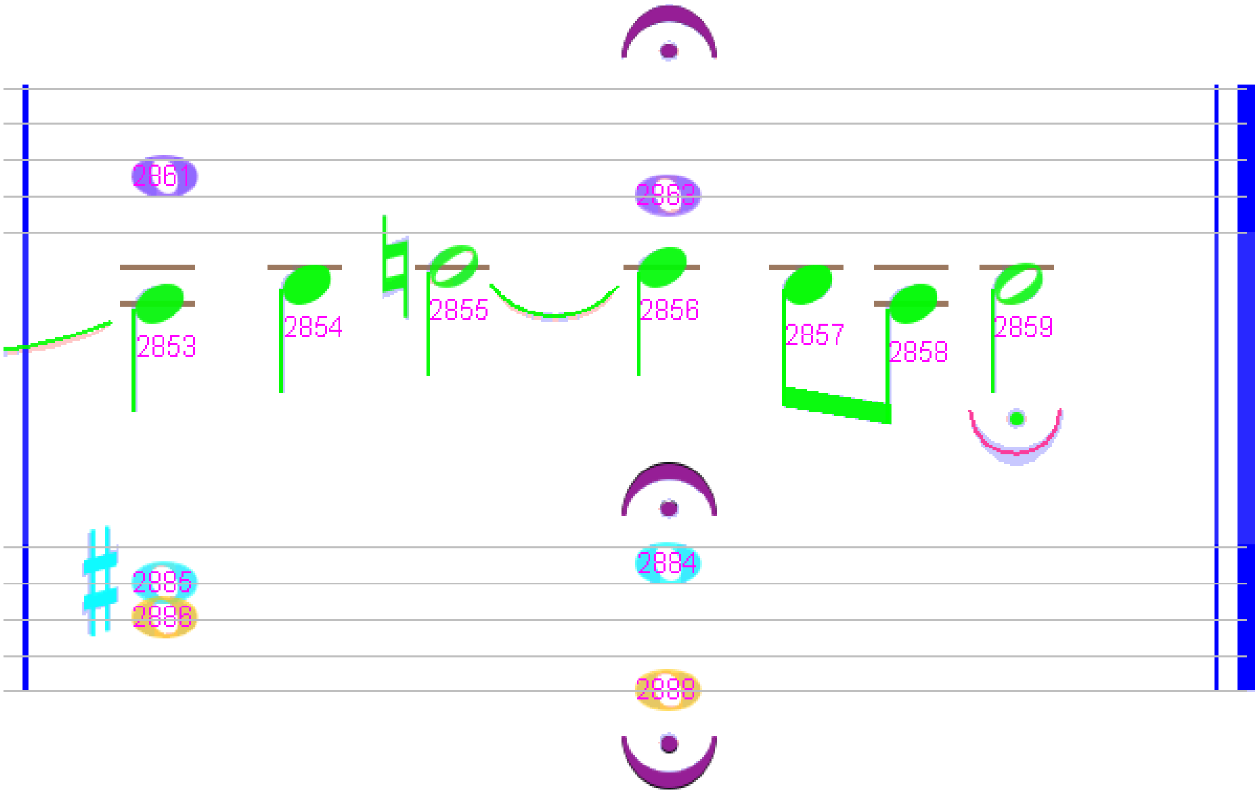
```
MeasureStack#6
    |0       |1/2     |3/4     |1       |5/4     |3/2     |7/4     |2
--- P1
V 1 |Ch#2756 |Ch#2758 |Ch#2760 =========|Ch#2762 |Ch#2763 =========|2
V 2 |Ch#2757 |Ch#2759 =========|Ch#2761 ==================|Ch#2764 |2
V 5 |Ch#3403 ==================|Ch#2792 =========|Ch#3334 =========|2
V 6 |Ch#2790 |Ch#2791 =========|Ch#2796 ===========================|2
```

Sheet #2, Measure #9



A bunch of unwanted items in the upper left corner. To be manually deleted.

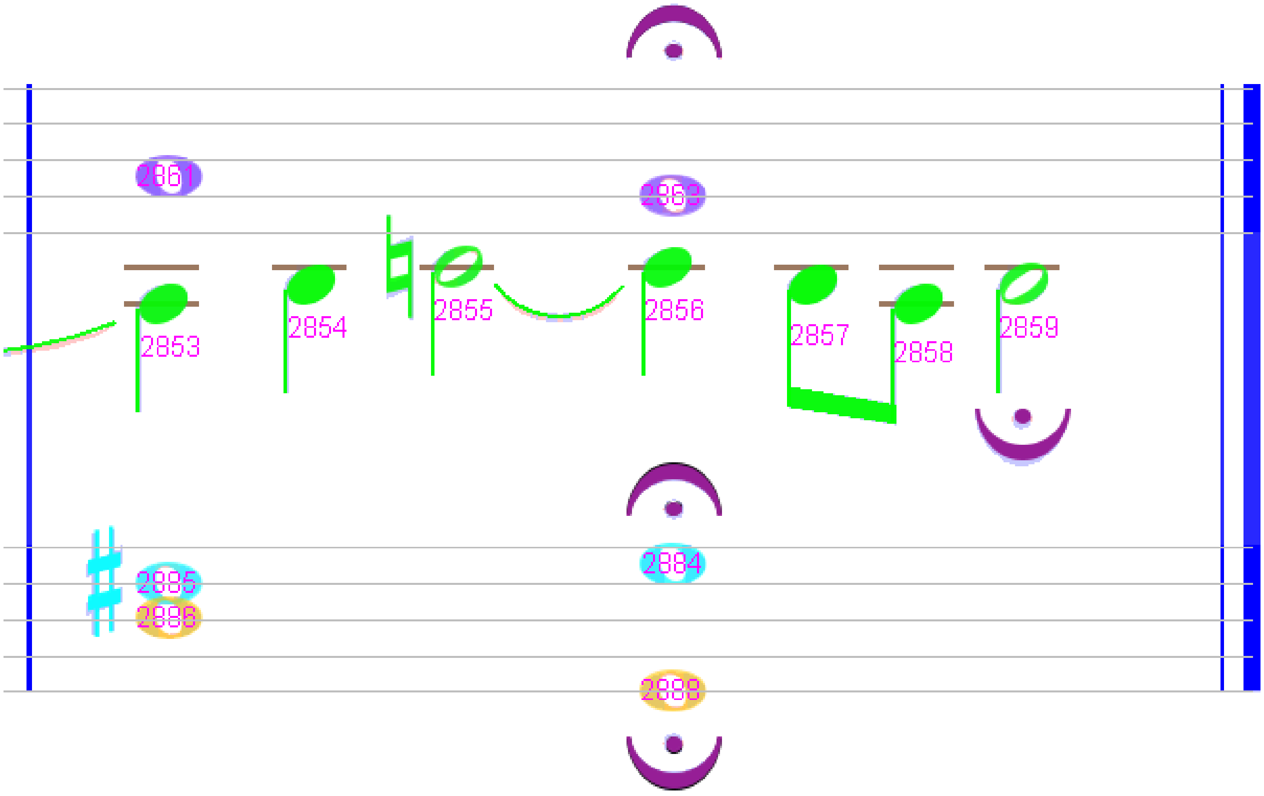
Sheet #2, Measure #17



This last measure in the sheet exhibits a missing fermata (mistaken for a slur and a staccato dot, located below chord #2859).

We delete both slur and staccato.

We then select the two underlying glyphs and, using the shape palette, we assign the shape FERMATA_BELOW to the compound glyph.



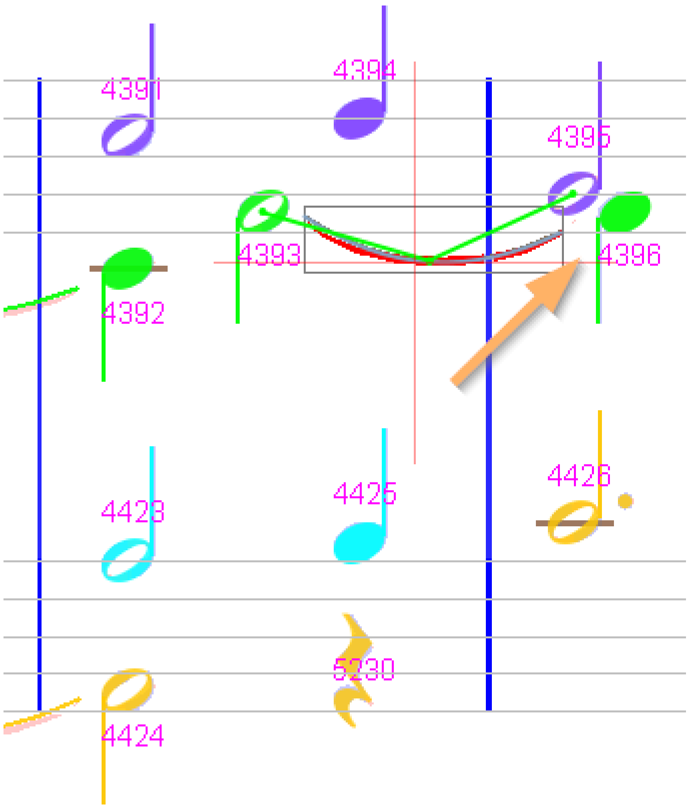
This completes our work on sheet #2.

Sheet #3

This is a new movement, governed by a 3/4 time signature.

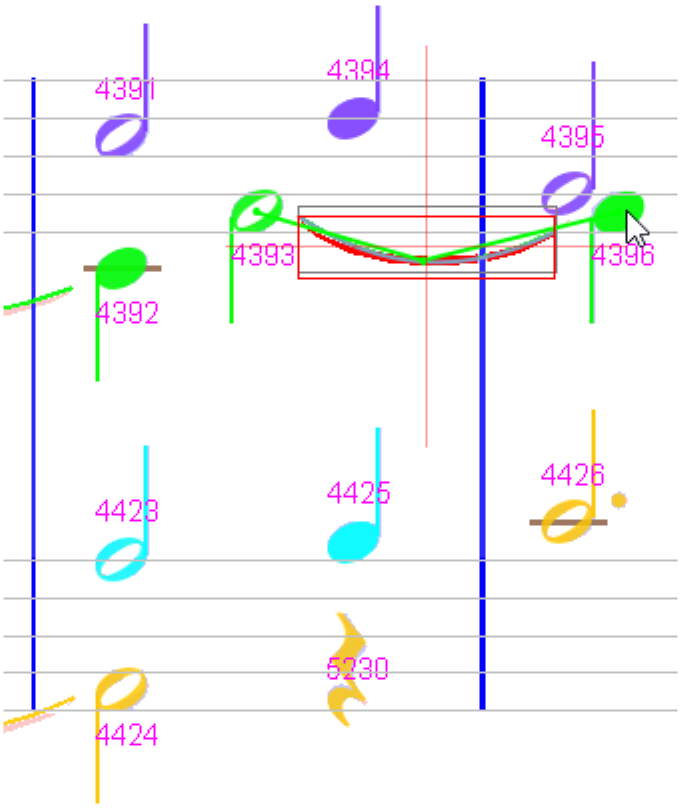
No pink measure in this page, but let’s have a closer look…

Sheet #3, Measure #11

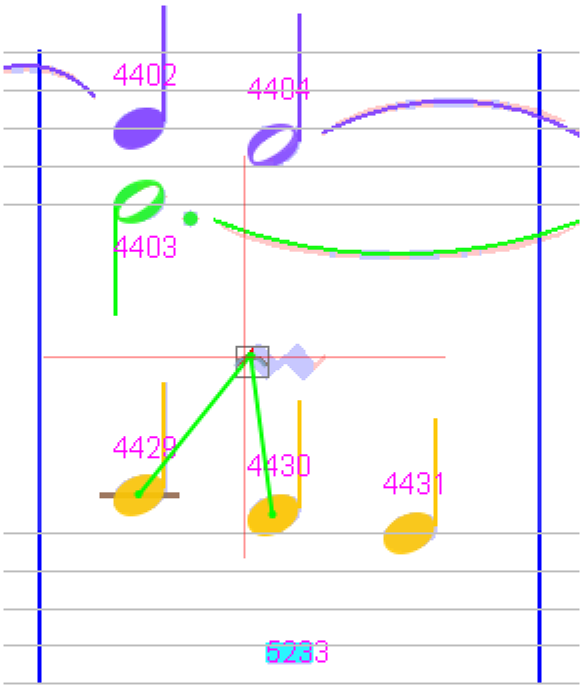


On the upper staff, we do have a slur but it is linked to the wrong note head on its right side, certainly because this head is closer to the slur end target.

So we drag from this slur to the head of chord #4396; this moves the slur-head relation off of chord #4395 head to chord #4396 head.



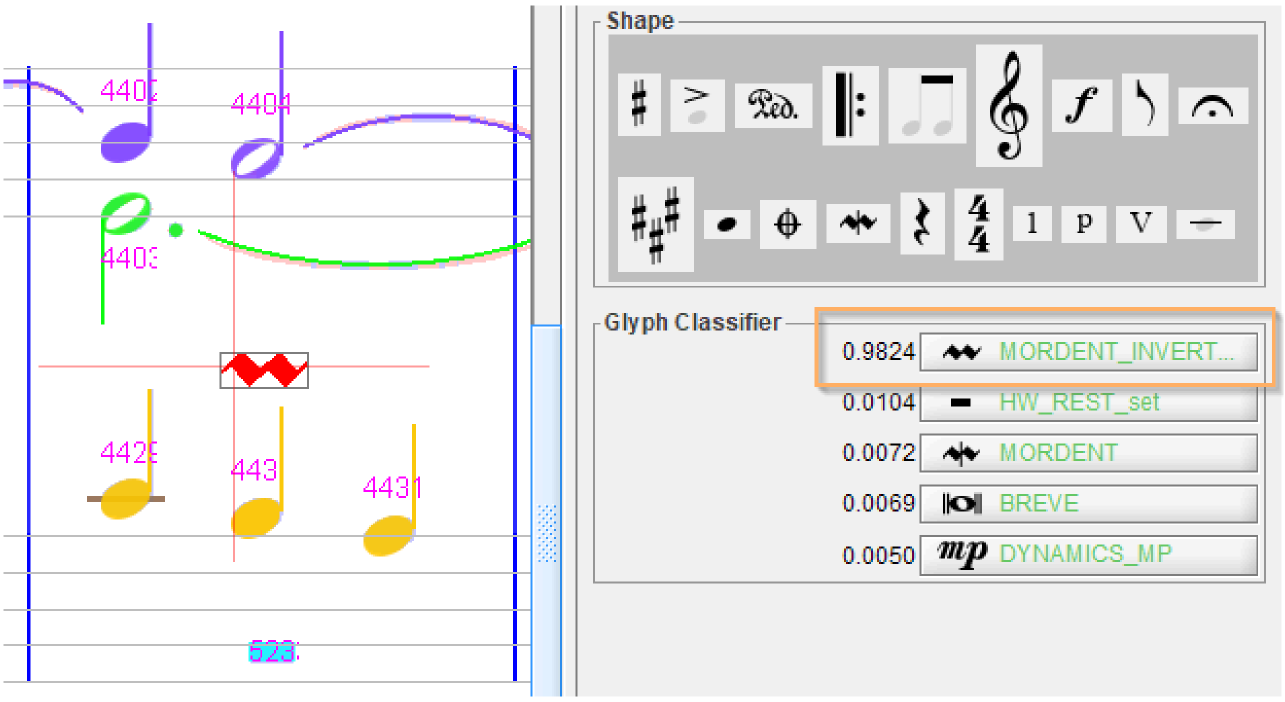
Sheet #3, Measure #14



Here we have a false tiny slur on an un-detected inverted mordent.

So we first delete the false slur.

Then we select the underlying glyph for the mordent.



0.9824 MORDENT_INVERT...
0.0104 HW_REST_set
0.0072 MORDENT
0.0069 BREVE
0.0050 DYNAMICS_MP

This triggers the glyph classifier which proposes the MORDENT_INVERT shape with a 0.9824 grade. We thus simply click on this top button.

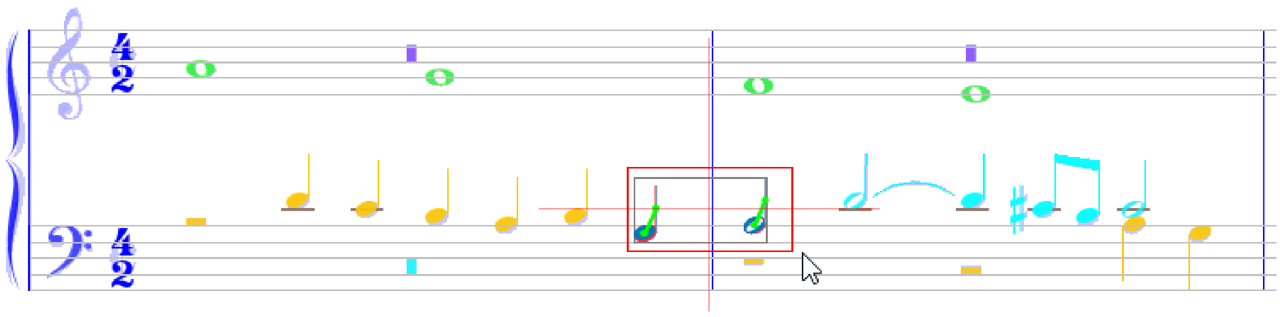
We have now completed sheet #3.

Sheet #4

As for sheet #1, we delete the common-cut time signatures and replace them with custom 4/2 signatures.

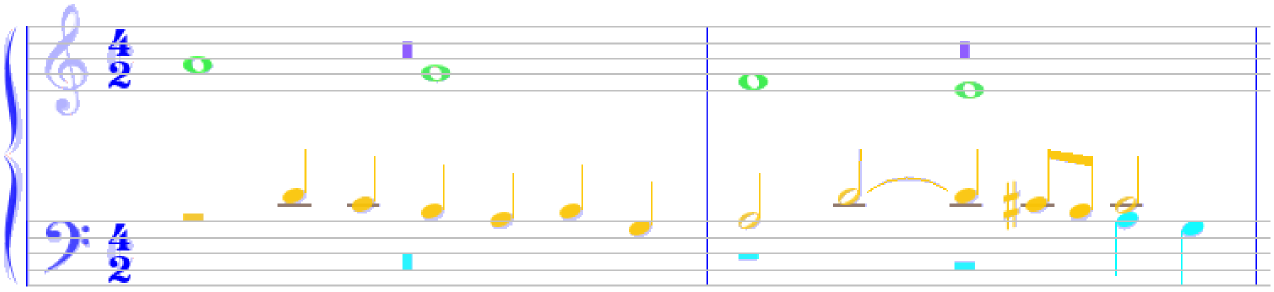
3 measures are left in pink.

Sheet #4, Measures #1 & #2



The first two measures are OK, but we would like to align their voices in the lower staff.

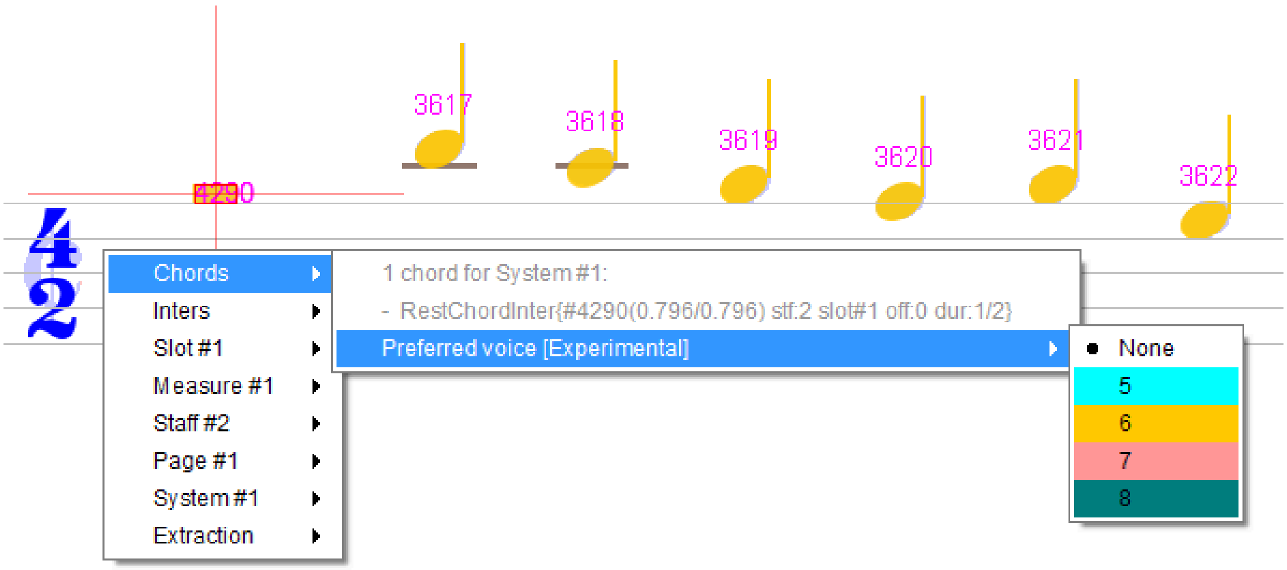
So, we select the last chord in measure #1 and the first chord in measure #2, and use the contextual `≡ Chords → Next in voice` to get:



And we do the same voice alignment action between measure #2 and measure #3.

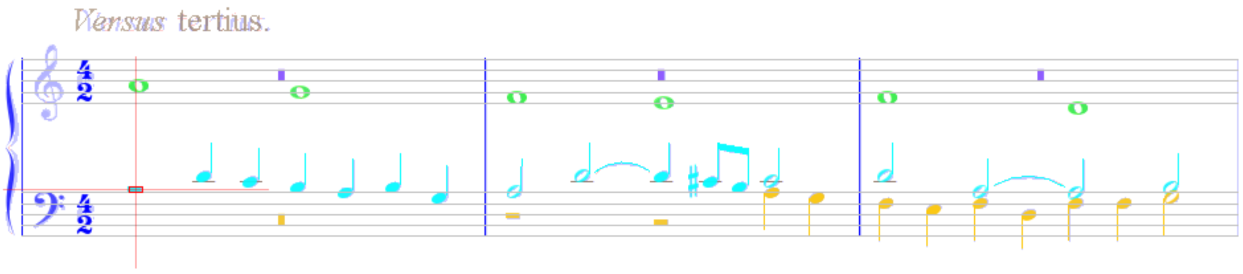
All voices are correctly aligned for this system, but we may not like the fact that the lower staff exhibits voice #6 (displayed in orange) **above** voice #5 (displayed in cyan). This is so because the Audiveris engine assigns the voices for measure-long rests first, and the “standard” voices only afterwards.

To swap these voices, we can use direct voice assignment as follows: We select (a head in) the first voice chord in the staff, for example chord #4290.



Then in the `≡ Chords` contextual menu, we select the “Preferred voice” directly for this chord.
We can see here that there was no voice assigned (“None”). We select the desired voice number in the menu, that is voice #5 (cyan color).

The engine then swaps voices for this measure (#1), and also in the following measures #2 and #3 to fulfill the “Next In Voice” relations we had just set.

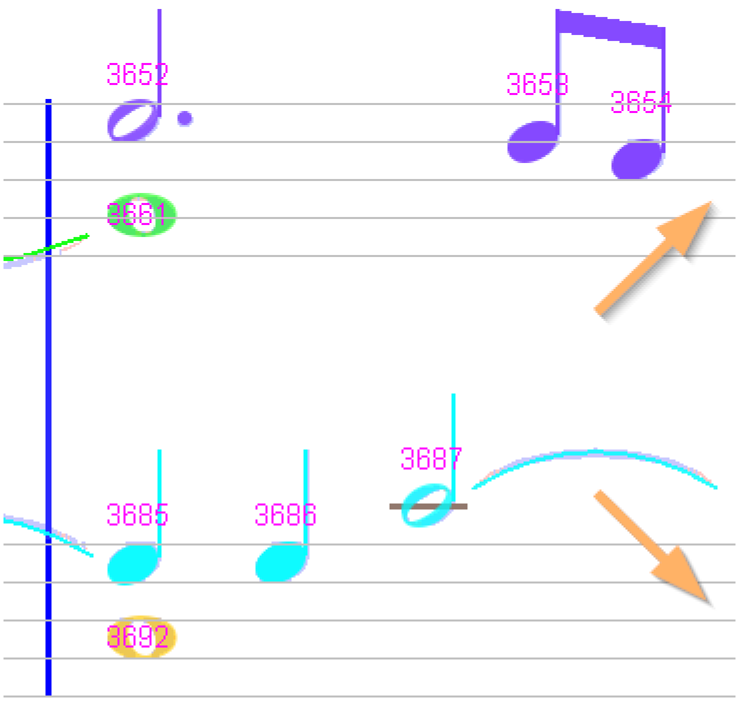
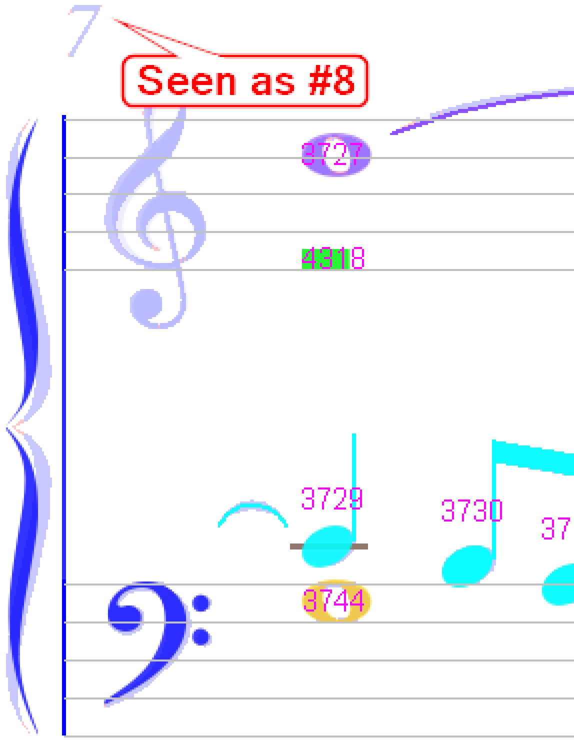


WARNING

This direct voice number assignment is still an experimental feature:

- Though it can be undone, it is a *hard coding*, hence less flexible than the use of relations like “Next in Voice” or “Separate Voices”,
- It is effective *only* for the starting chord of a voice in its measure. Setting the voice number for another chord in the voice will have no effect, and in particular cannot be used as a means to link or unlink voices.

Sheet #4, Measures #7 & #8

Half-measure #7a (seen as #7)	Half-measure #7b (se
 3652 3653 3654 3661 3687 3685 3686 3692	 7 Seen as #8 3727 4318 3729 3730 37 3744

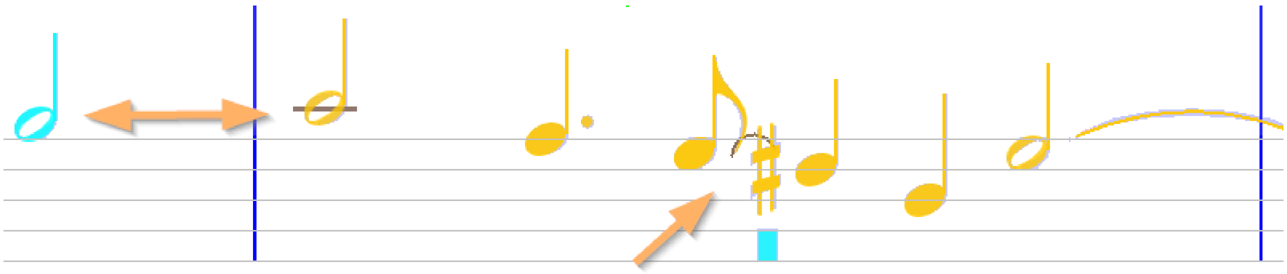
Here we have the strange layout of a measure that ends system #2 and starts system #3. Notice that system #2 ends with no barline, as opposed to the other systems. Each of these “half-measures” has a duration of 1.

Audiveris can see these only as separate measures, numbered #7 and #8 respectively.

NOTE

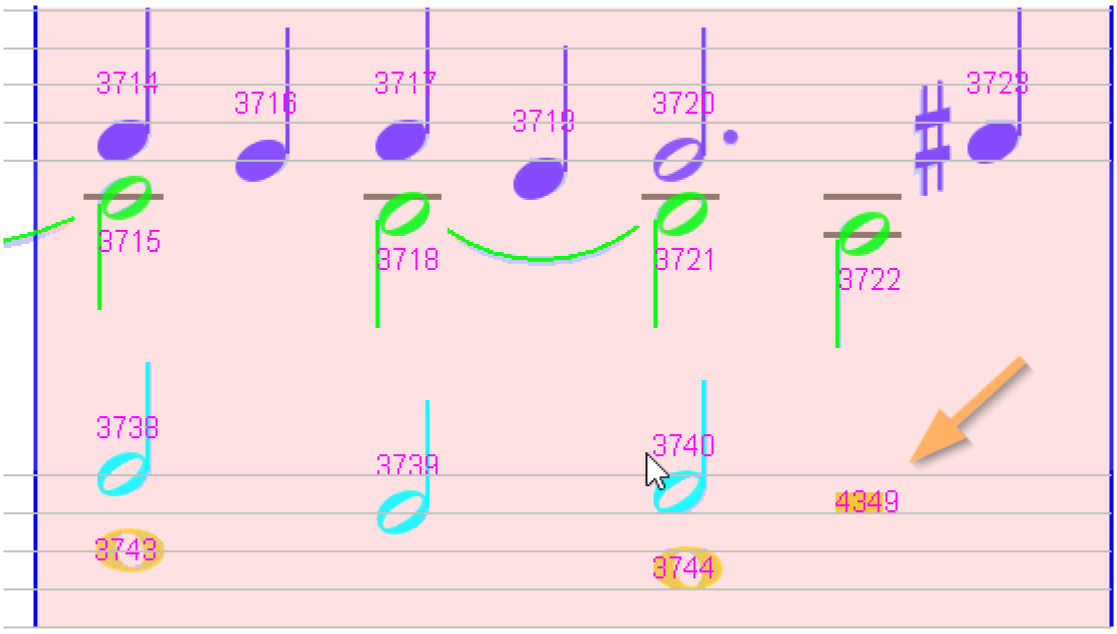
Unfortunately, this shifts by 1 all the following measure numbers in this movement, and there is nothing yet we can do to fix this…

Sheet #4, Measure #9



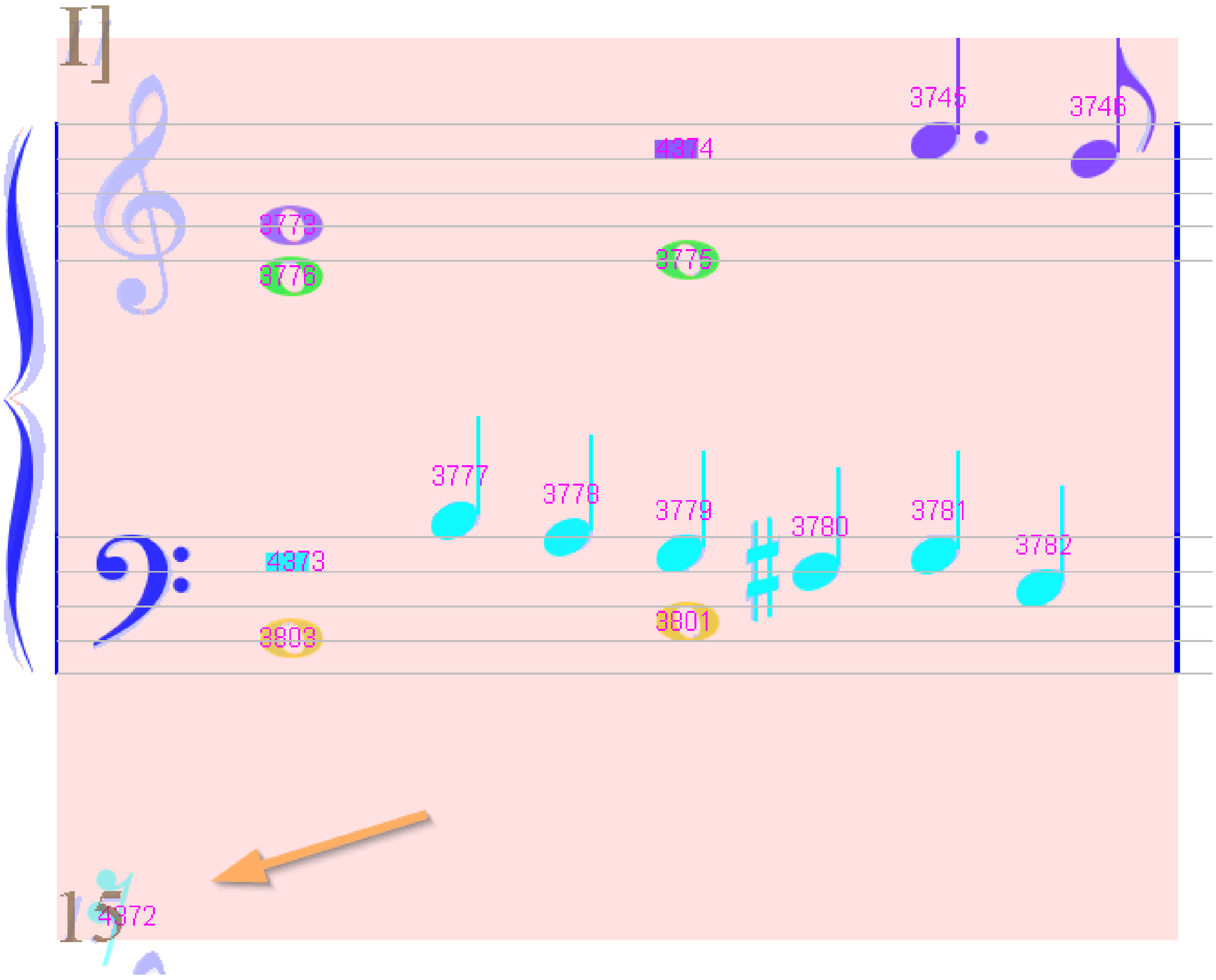
On the lower staff, we align voices between measure #8 and this measure #9.
We also delete a false slur in the middle of this staff.

Sheet #4, Measure #11



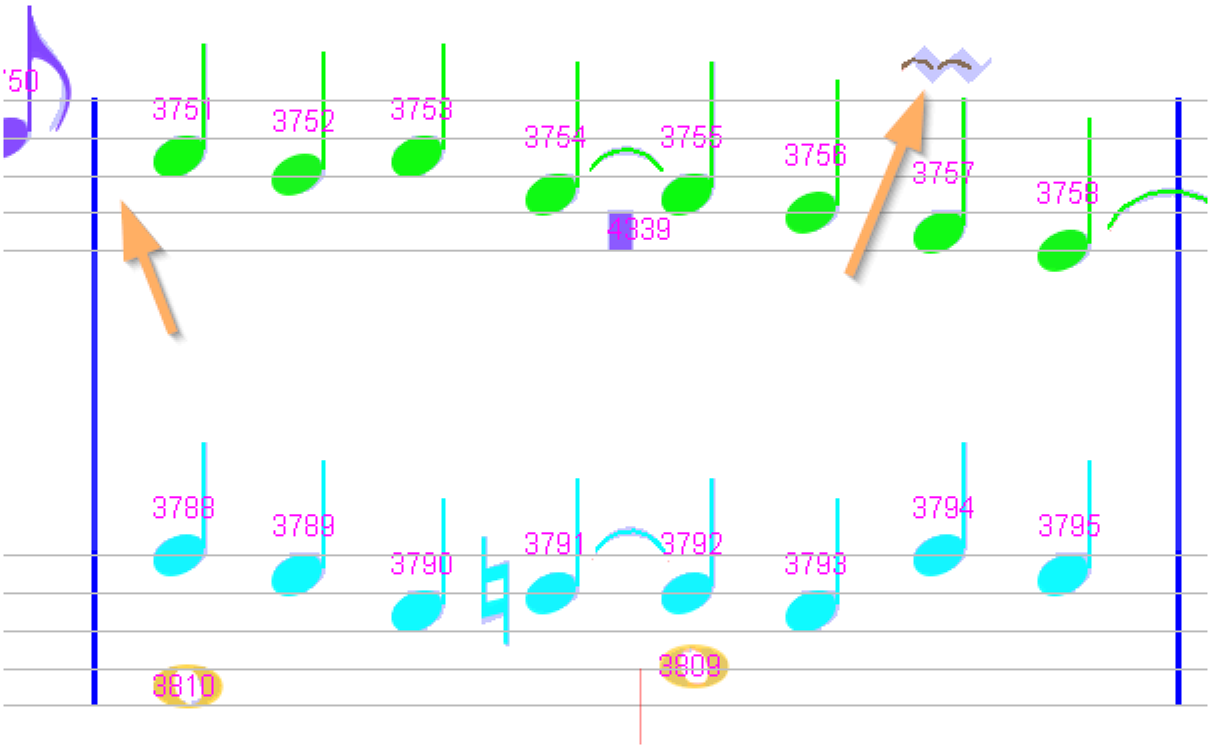
A measure in pink because of a too long voice.
This is due to the final half rest, which should belong to the voice of the preceding half chord, rather than the preceding whole chord.

Sheet #4, Measure #12



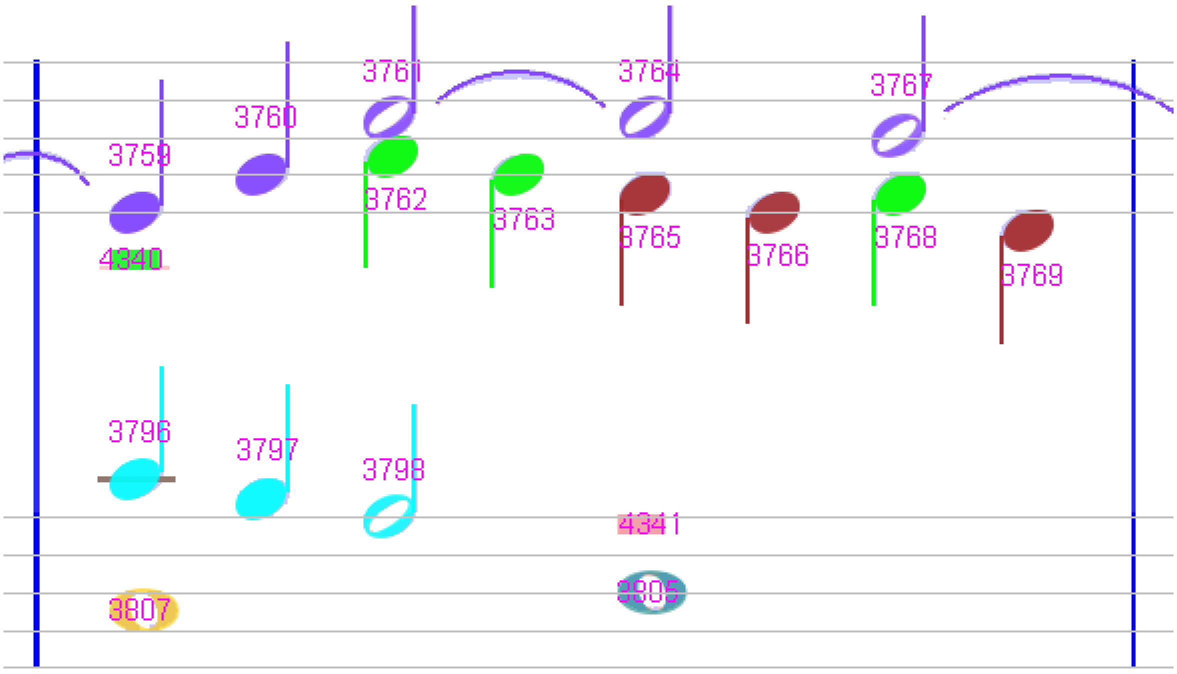
We can see the pink background going very low under the measure staves. This reveals the presence of an undesired 16th rest, to be deleted.

Sheet #4, Measure #13



We align voices with the preceding measure, delete two false small slurs and replace them by an inverted mordent.

Sheet #4, Measure #14



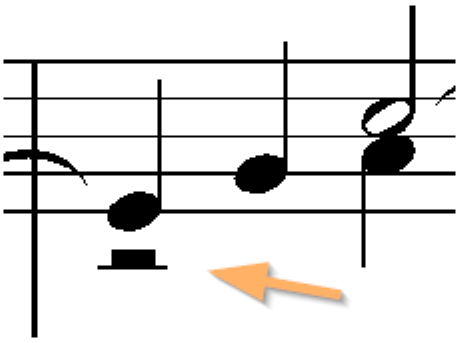
This measure is not in pink because no voice lasts longer than the measure duration (4/2), however the rhythm is strange:

- On the upper staff, we have 3 voices, while 2 would be enough.
- On the lower staff, we have 4 voices, while 2 would be enough.

```
MeasureStack#14
    |0       |1/4     |1/2     |1       |5/4     |3/2     |2
--- P1
V 1 |Ch#3759 |Ch#3760 |Ch#3761 |Ch#3764 =========|Ch#3767 |2
V 2 |Ch#4340 ==================|Ch#3762 |Ch#3763 |Ch#3768 |7/4
V 3 |........|........|........|Ch#3765 |Ch#3766 |Ch#3769 |7/4
V 5 |Ch#3796 |Ch#3797 |Ch#3798 |........|........|........|1
V 6 |Ch#3807 ==================|........|........|........|1
V 7 |........|........|........|Ch#4341 ==================|2
V 8 |........|........|........|Ch#3805 ==================|2
```

The measure strip shows that chord #4340 is considered as a whole rest. Also clicking on this rest shows WHOLE_REST in the `Inter` board.

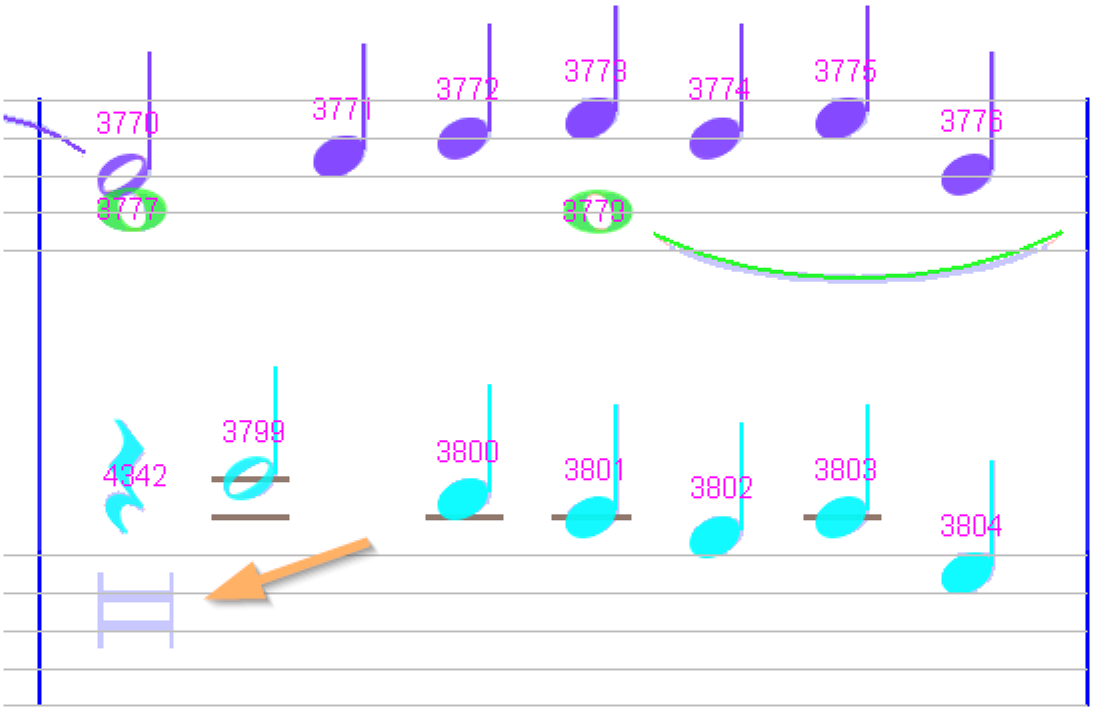
However, a look at the binary image gives a different result:



Once again, this rest was mistaken for a whole rest because of its vertical location with respect to the related staff. So, we delete it and manually reassign the underlying glyph as a HALF_REST.

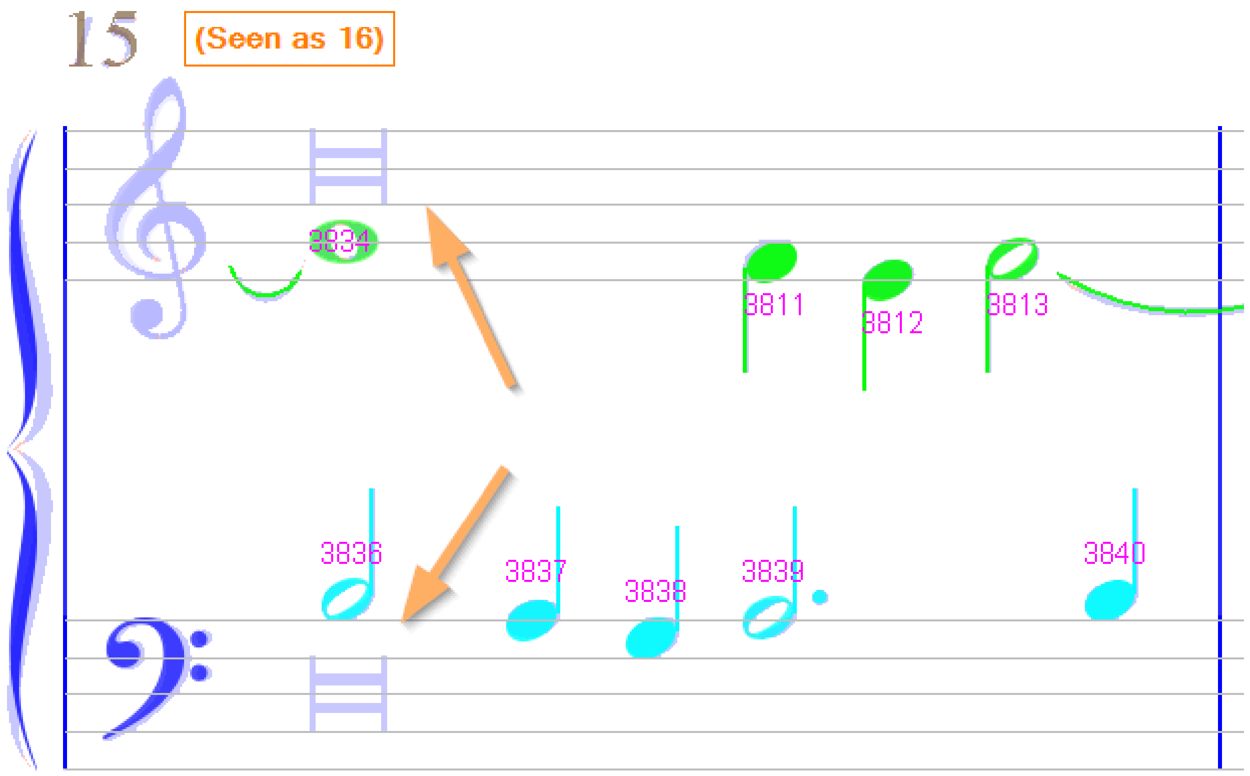
This immediately fixes the rhythm on both upper and lower staves.

Sheet #4, Measure #15

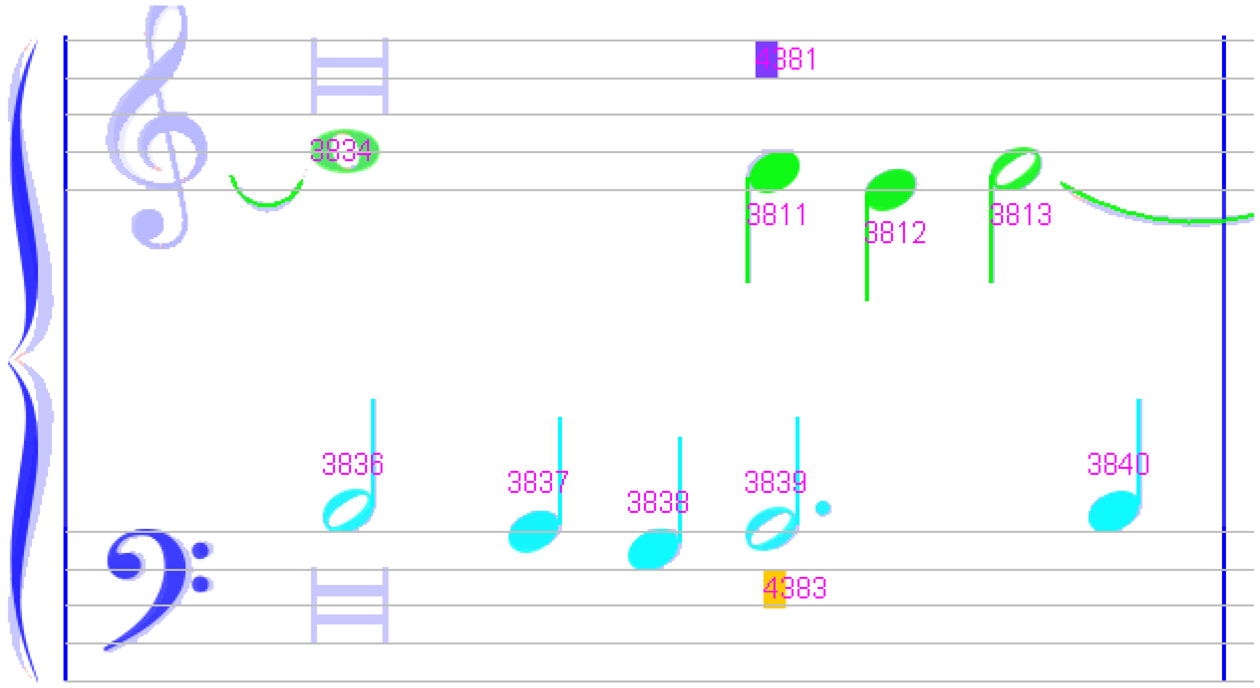


The sign at beginning of the lower staff is an old sign for a multi-measure rest, a kind of entity not recognized by Audiveris.
Since the measure duration is 2, we can manually insert a breve rest at the measure center, and voice-link it (using “Next in Voice”) with the corresponding whole note in the previous measure.

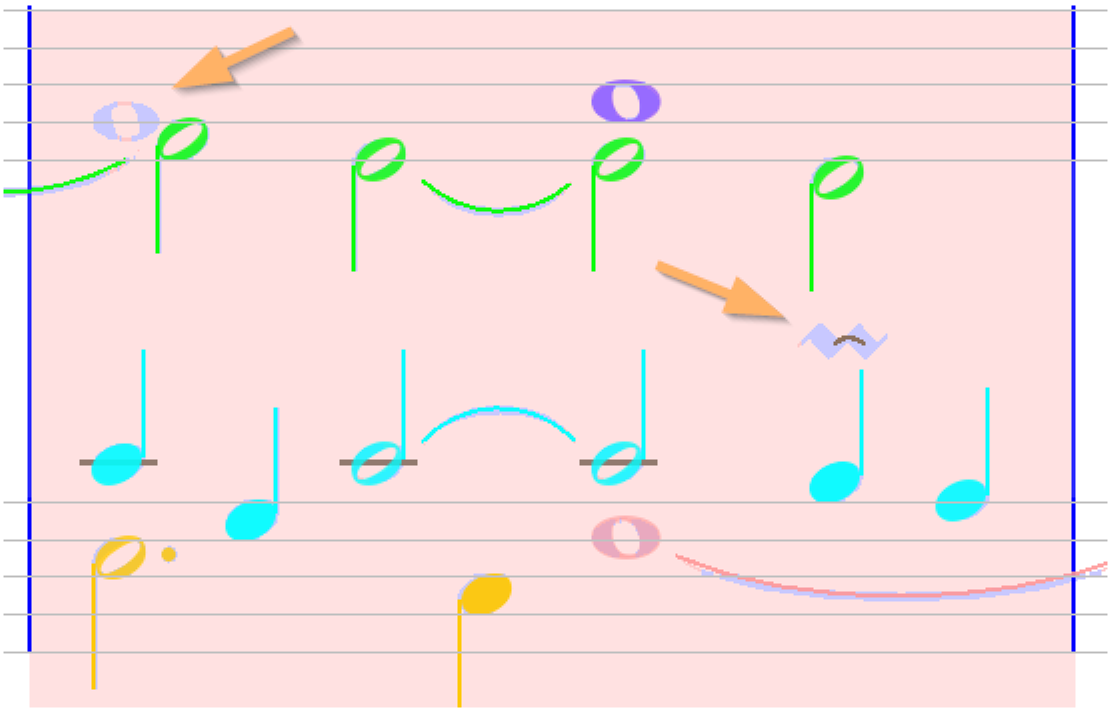
Sheet #4, Measure #16



Here again, we can manually insert breve rests in the upper and lower staves. And assign the correct voice number (5) for starting chord #3836.

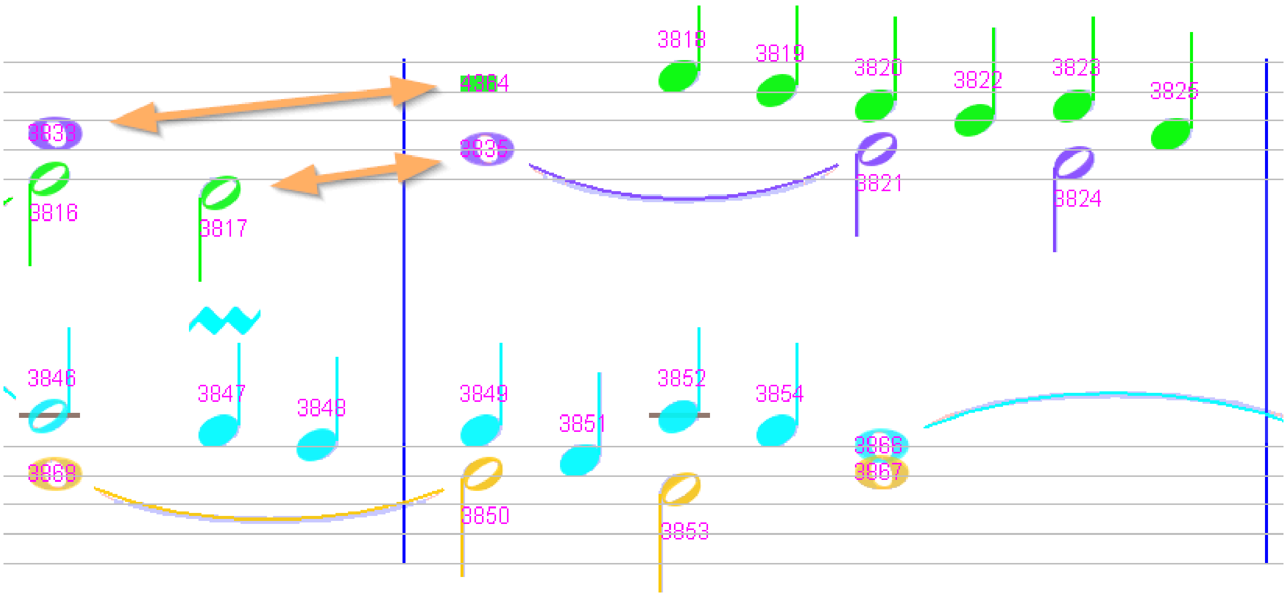


Sheet #4, Measure #17



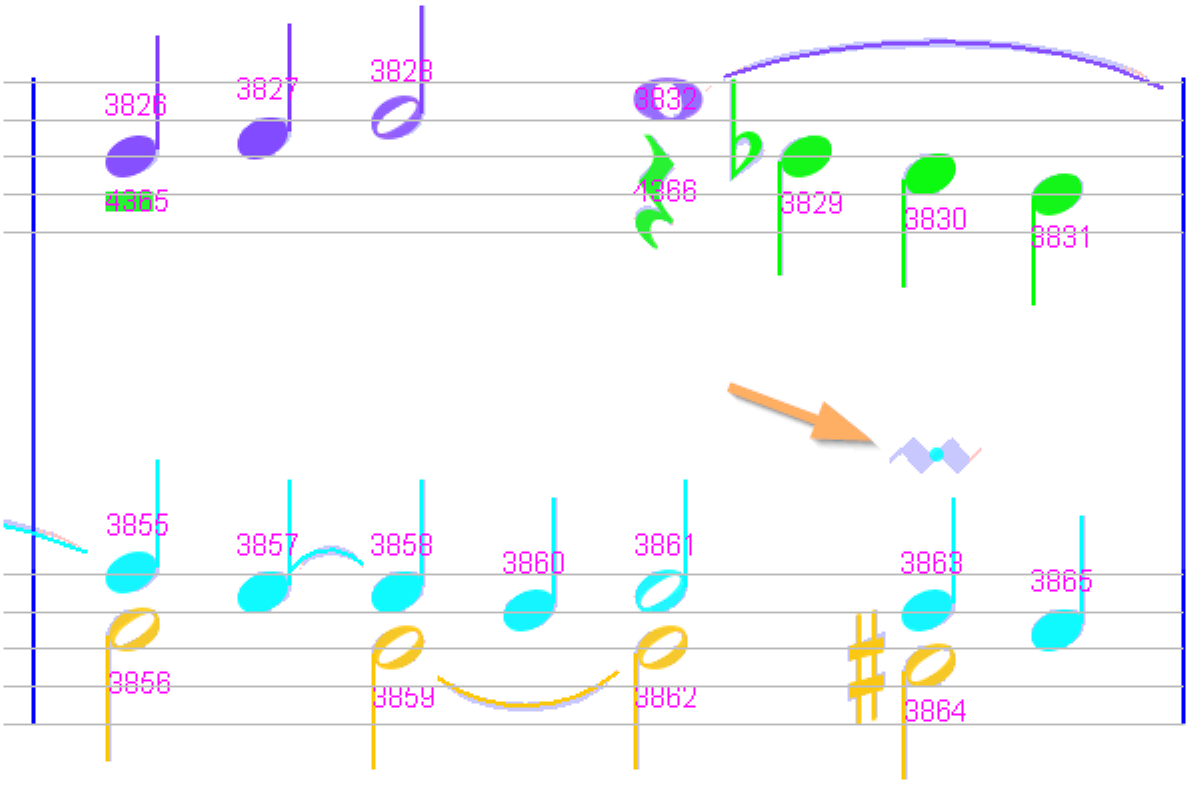
There is an un-recognized whole note in the upper left corner (to be inserted), a false slur (to be deleted) and an un-recognized inverted mordent (to be assigned from its underlying glyph).

Sheet #4, Measure #18



The measure rhythm is correct, but in the upper staff we can align voices with the previous measure.

Sheet #4, Measure #19



Finally a false staccato dot (to be deleted) and yet another inverted mordent to be assigned from its underlying glyph.

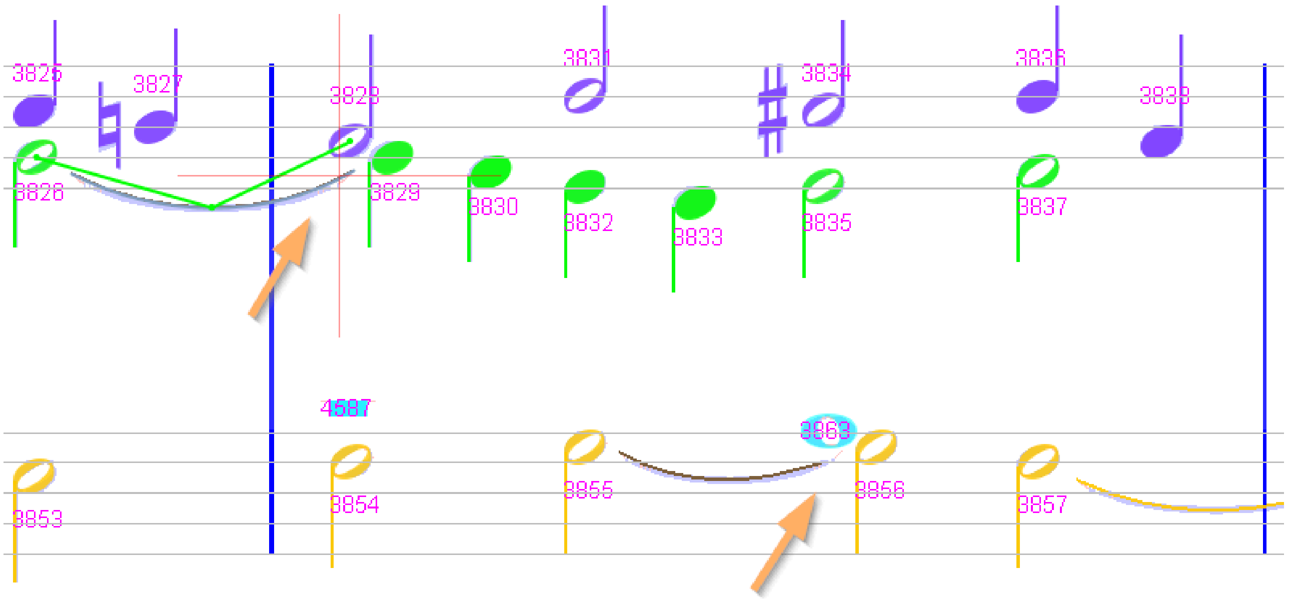
This completes our work on sheet #4.

Sheet #5

This sheet #5 continues the movement begun with sheet #4.

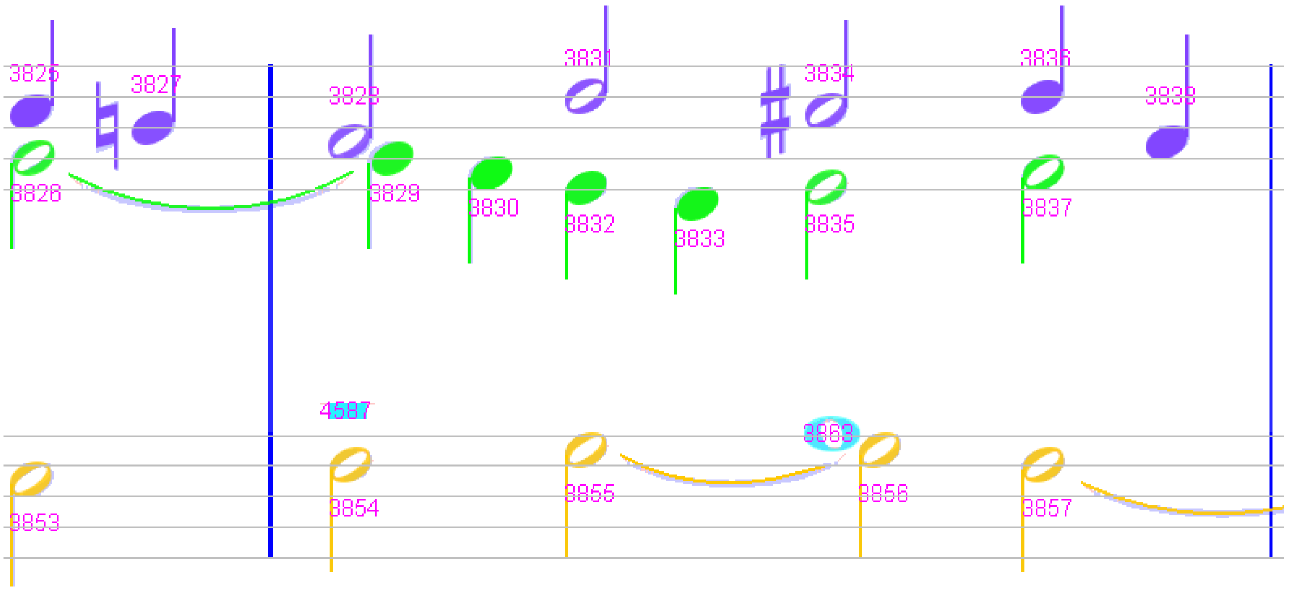
We update the page rhythm. There are two measures left in pink in this sheet plus a few others to fix.

Sheet #5, Measure #2

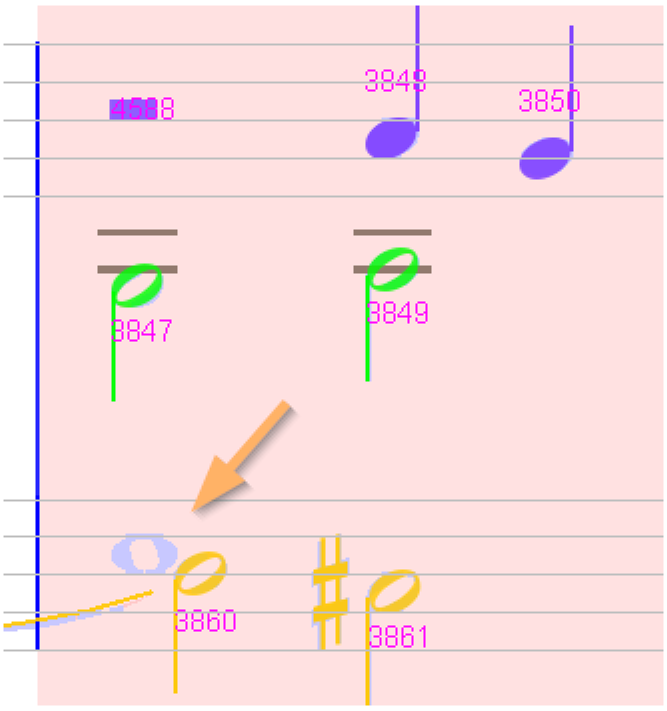


We have two slurs that should be recognized as tie slurs but are not (and are thus displayed in black, instead of a voice color).
The reason is that both are connected to the wrong head on their right side.

To fix this, we drag from each slur to the correct target head and release the mouse.
Each slur is now recognized as a tie slur and its color turns to the color of its underlying voice.

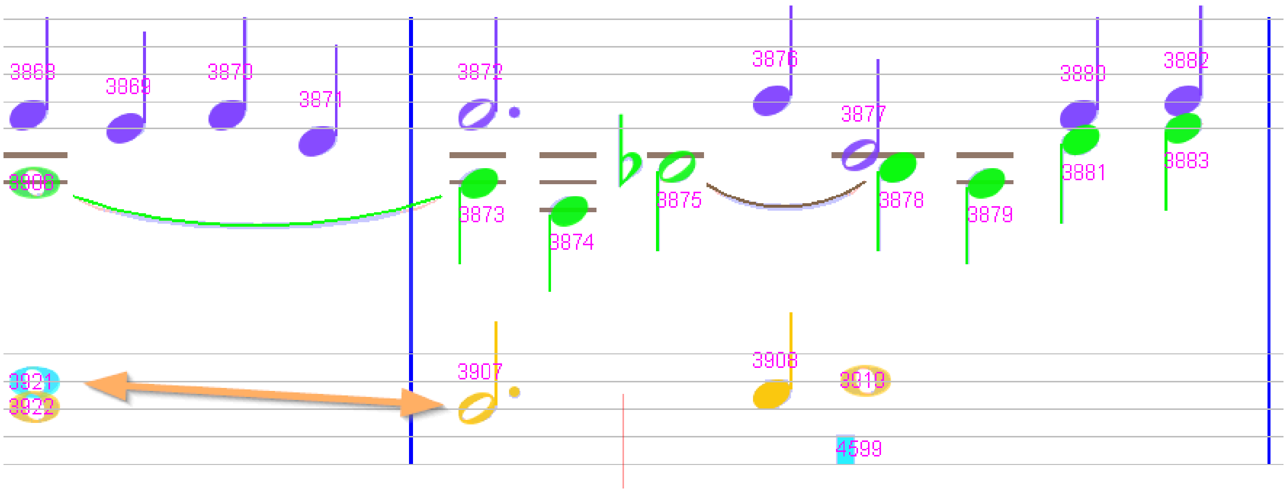


Sheet #5, Measure #4



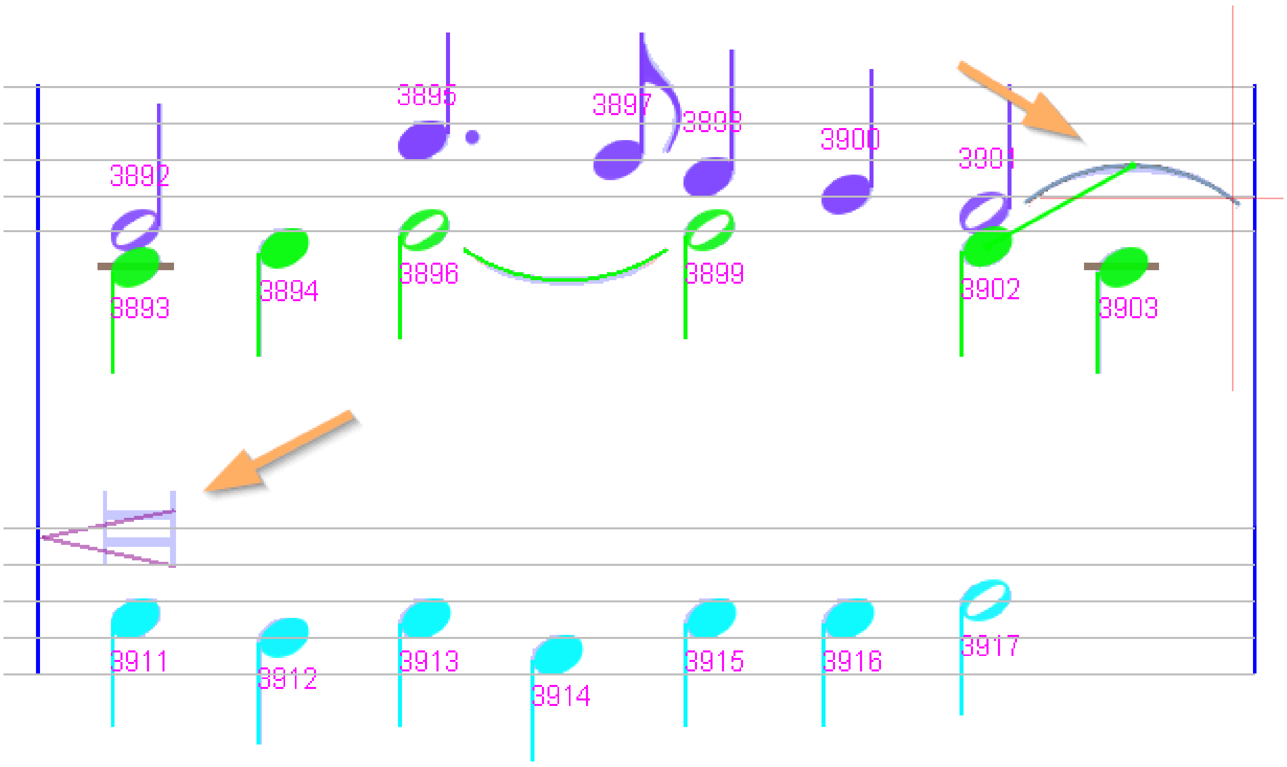
A missing whole head (in this half measure).

Sheet #5, Measure #6



In the lower staff, voices need to be aligned with the previous measure, for example by setting a “Next in Voice” relation between chords #3921 and #3907.

Sheet #5, Measure #8



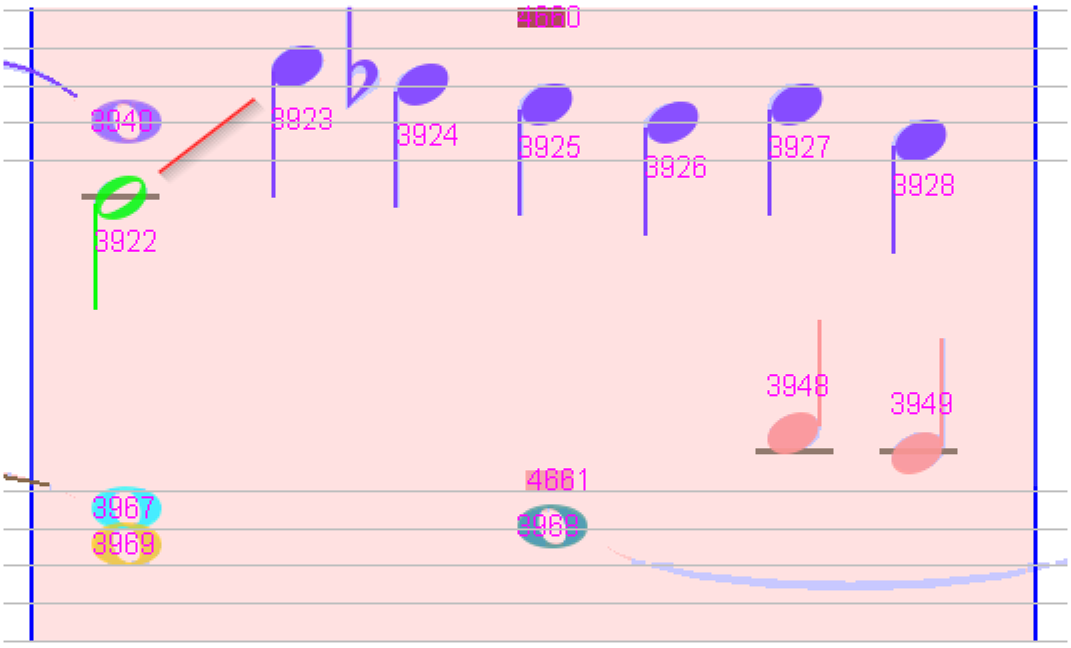
Lower staff: A false crescendo to delete and a breve rest to insert.

Upper staff: a slur that should be a tie. The reason is this slur is not connected on its left side to the correct head, it must be moved from #3902 to #3901

Sheet #5, Measure #9

Again a breve rest to be manually inserted.

Sheet #5, Measure #10



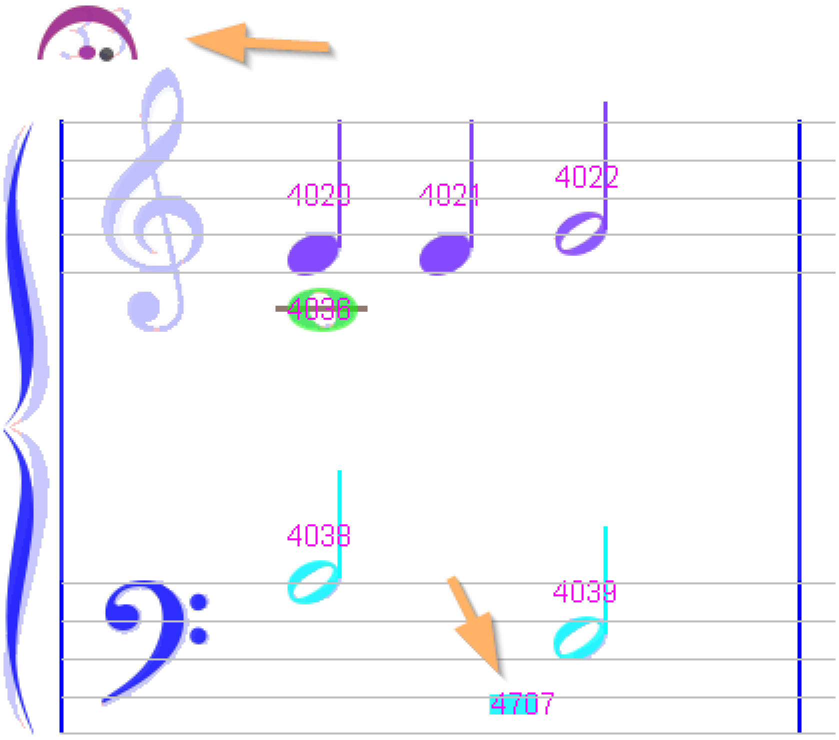
The pink measure is due to an incorrect voice connection in upper staff.

Aligning voices correctly in the upper staff has the side effect of aligning them correctly in the lower staff as well.

Sheet #5, Measure #12

In the upper staff, voices need to be swapped by connecting them to the corresponding voice in the previous measure.

Sheet #5, Measure #17



There is a false fermata at (the beginning! of) the upper staff. We delete it.

```
MeasureStack#17
    |0       |1/4     |1/2     |3/2     |2
--- P1
V 1 |Ch#4020 |Ch#4021 |Ch#4022 |........|1
V 2 |Ch#4036 ==================|........|1
V 5 |Ch#4038 =========|Ch#4707 |Ch#4039 |2
```

There is also a problem with the measure length.

In fact, it looks like a “half” measure which continues another “half” measure at the end of the previous system with no ending barline.

For the Audiveris data model, these are two separate measures. And since neither of them lasts longer than 4/2, they are not flagged as abnormal pink measures.

However, the duration of voice #5 in the second half measure is 2, while we would expect 1.
This is because of the whole rest chord #4707, which is inserted between the half chords #4038 & #4039 in a single voice.

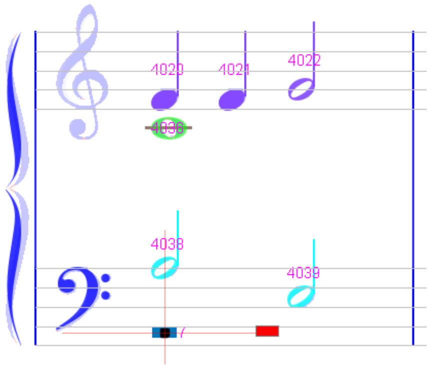
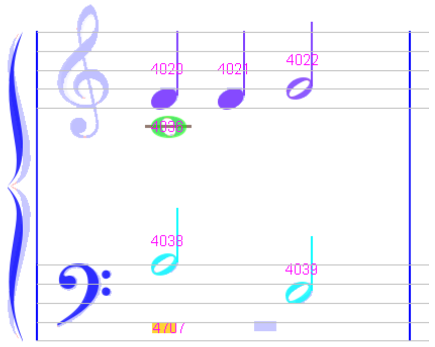
We can try to force voice separation between the whole rest chord #4707 and the surrounding half chords. This results in a message like:

```
Measure{#17P1} No timeOffset for RestChordInter{#4707(0.796/0.796) stf:10 slot#3 dur:1}
```

In other words, the engine fails to determine the time offset of the whole rest chord #4707, alone in its voice. This is why the measure background turns pink.

In fact, we can consider that the original score here is wrong.
The whole rest, as it is located here in the middle of the measure, should be considered as a measure-long rest, but this would not be consistent with the other measures on this page.

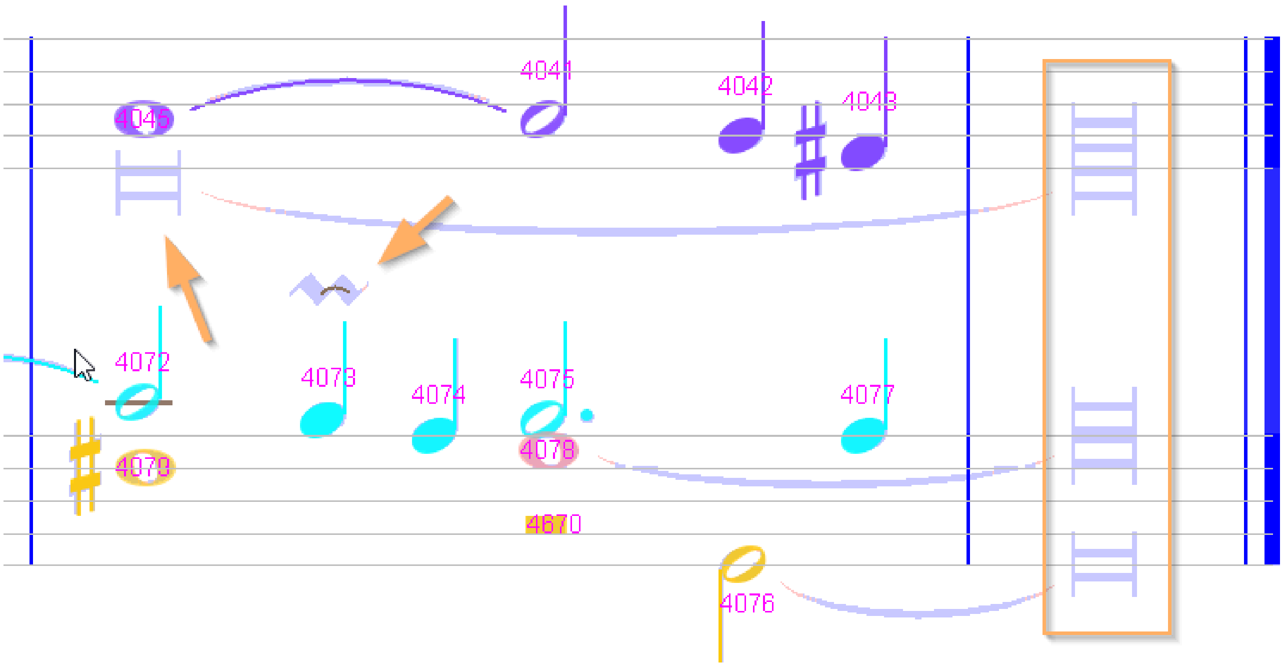
So a workaround is to actually *shift* this badly located rest to the left, at the beginning of the measure.
To do so, via a double-click, we put the symbol into edit mode, and drag its handle to the left.

Shift editing	End result
	

Sheet #5, Measure #19

Again a breve rest to be manually inserted.

Sheet #5, Measure #20



We have a breve rest to insert on the upper staff. Plus a small slur to delete and an inverted mordent to insert.

Apart from that, the role of a slur that connects to a rest is not obvious. Perhaps we can just ignore these slurs, and the next measure as well.

This completes our work on sheet #5,
and concludes the whole editing session.

Completion

In this editing session, we have focused on pure musical content, that is essentially the rhythm aspect (chords organized in voices and time slots).

Text

There are a few textual elements in this book, most of them are located on the first sheet.
We can see a mix of several languages the OCR had to deal with.

Note, for visual check:

- the original text is easier to read in the binary view in physical mode ♫
- while the OCR output is easier to read in logical mode ♫

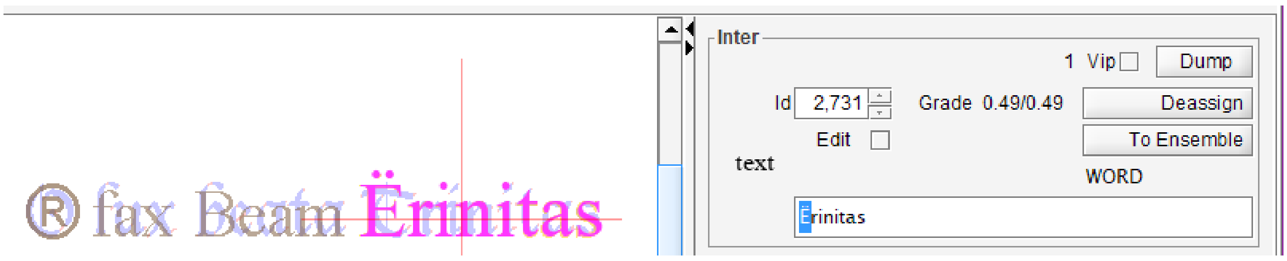
Here are all the textual elements found in this book (we ignore text that represents a measure number)

Original sentence	OCR output	Comments
O lux beata Trinitas	® fax Beam Ërinitas	Latin, gothic? font, OCR totally wrong
Der du bist drei in Einigkeit.	Der du *bist drei* in *Einigkeit.*	German, OCR mix of italic & standard
Restitution par P. Gouin.	Restitution par P. Gouin.	French
Johann Friedrich Alberti	Johann Friedrich Alberti	
(1642-1710)	(1642-1710)	
Versus primus.	*Versusprimus.*	Latin, OCR merged as one word
Versus secundus.	*Versus* secunafus.	Latin, OCR mistake on “secundus”
Versus tertius.	*Versus* tertius.	Latin, OCR mix of italic and standard
© Les Éditions Outremontaises - 2018	© Les Editions Outremontaises - 2018	French, OCR took ‘É’ for ‘E’

If we except the book title “O lux beata Trinitas” which OCR totally failed to recognize, probably because of its exotic font, the raw Tesseract OCR results are pretty good.

We can manually correct the few mistakes, via the `Inter` board.
This must be done word by word, by selecting the word and modifying its content (typing new characters and validating by pressing the `Enter` key).

Note that the current Audiveris UI let us modify only the textual **content** of any word, not its **font** characteristics (such as font name, italic, bold, etc).



1

Id 2,731

Grade 0.49/0.49

Edit □

text

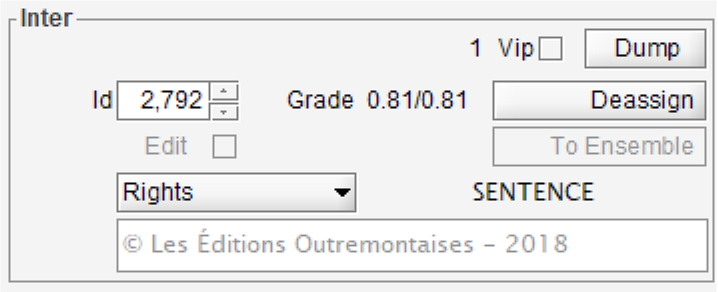
Ërinitas

For manual text editing, text **content** is modified at the *word* level whereas text **role** is modified at the *sentence* level.

In this book, nearly all sentences had their role correctly assigned by the OMR engine heuristics. An exception is “© Les Éditions Outremontaises - 2018” whose role appears to be “UnknownRole”.

To fix this, we select one word of this sentence and, in the `Inter` board, click on the `To Ensemble` button (as you know, a sentence is modeled as an ensemble of words).

We can then choose the proper sentence role (“Rights”):



1

Id 2,792 Grade 0.81/0.81

Rights SENTENCE

© Les Éditions Outremontaises – 2018

MusicXML export

We can export to MusicXML via the `Book → Export book` menu item. This gives:

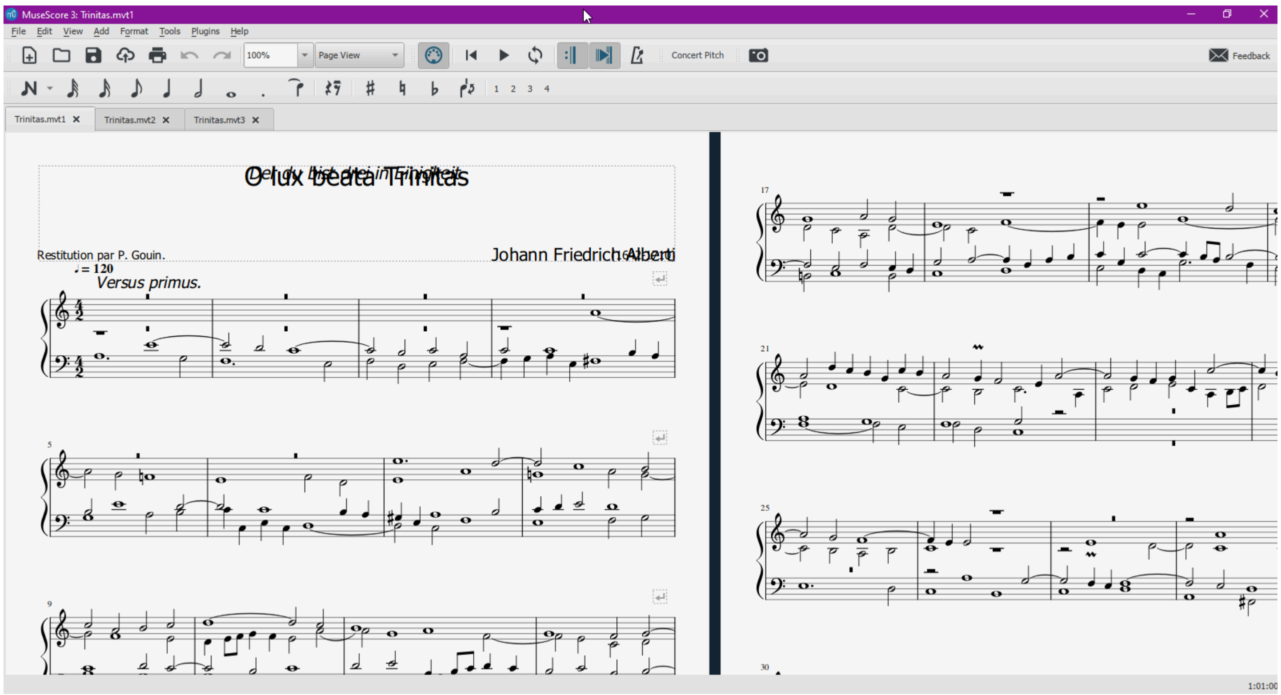
```
[Trinitas] Exporting sheet(s): [#1#2]
[Trinitas] Score Trinitas.mvt1 exported to D:\soft\cases\Issue-481\Trinitas.mvt1.mxl
[Trinitas] Exporting sheet(s): [#3]
[Trinitas] Score Trinitas.mvt2 exported to D:\soft\cases\Issue-481\Trinitas.mvt2.mxl
[Trinitas] Exporting sheet(s): [#4#5]
[Trinitas] Score Trinitas.mvt3 exported to D:\soft\cases\Issue-481\Trinitas.mvt3.mxl
```

This book was exported as 3 separate .mxl files, one per detected movement.

Plugin

If we have configured a few plugins to external programs such as MuseScore or Finale, clicking on the desired plugin would:

1. Export an updated version of the 3 movements files, if needed,
2. Launch for example MuseScore on the 3 movement files.



We can then for example listen to MuseScore playback on these movements. This is an additional convenient way to check transcription results.

Can’t Take My Eyes Off of You

This case is one of the examples discussed on issue #33 posted on Audiveris forum, dedicated to the support of drum notation.

Input file available here

Main score specificities:

- Book of 3 sheets
- Drum notation on 5-lines staves and 1-line staves
- Multiple-measure rests
- Measure repeats

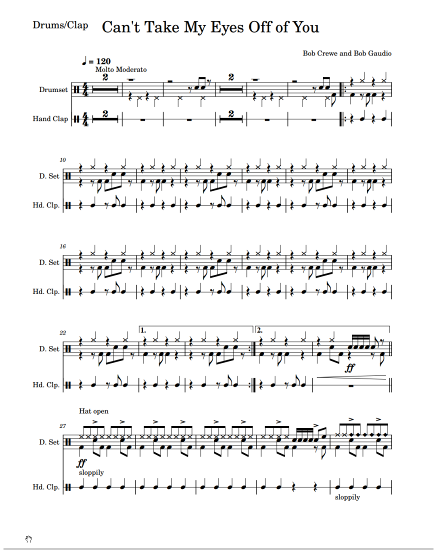
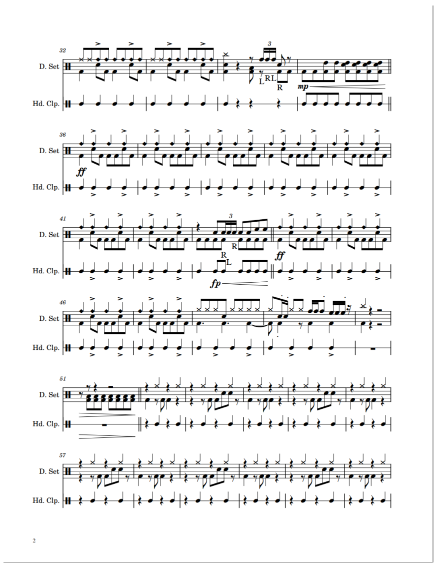
Main UI actions:

- Selection of proper book parameters for drum notation
- Customization of drum-set mapping
- Fix missing rests in original input
- A few symbols to fix
- Editing of logical parts

TABLE OF CONTENTS

Input

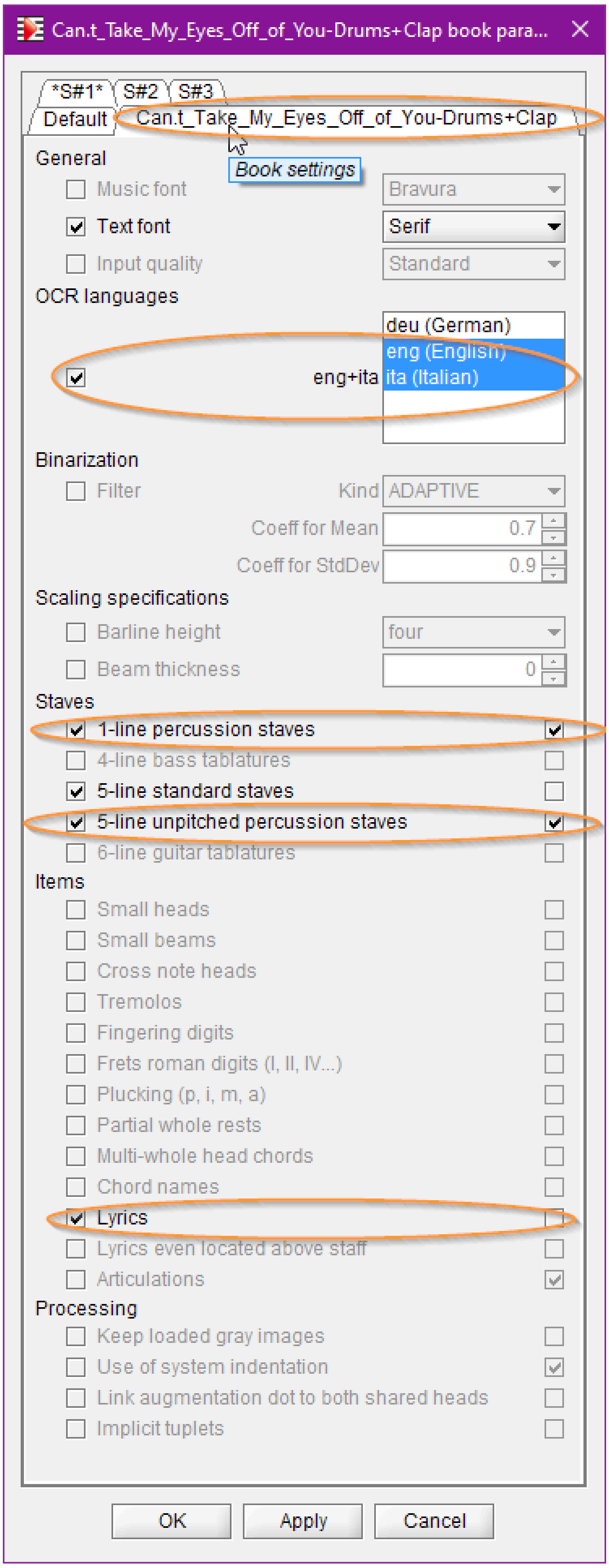
> If you are reading this handbook via a web browser, you can ask the browser to display each of these sheet images in a separate tab.

Sheet ID	Input image
Sheet#1	 Drums/Clap Can't Take My Eyes Off of You Bob Crewe and Bob Gaudio ♩= 120 Molto Moderato Drumset Hand Clap D. Set Hd. Clp. 1. 2. ff Hat open sloppily
Sheet#2	

Sheet ID	Input image
Sheet#3	

Book parameters

Based on a first look at the input score, we can select the proper book parameters, via the `Book → Set book parameters` dialog:



Can.t_Take_My_Eyes_Off_of_You-Drums+Clap book para...

S#1 S#2 S#3

Default | Can.t_Take_My_Eyes_Off_of_You-Drums+Clap

Book settings

General

- □ Music font — Bravura
- ☑ Text font — Serif
- □ Input quality — Standard

OCR languages

- ☑ eng+ita — deu (German), **eng (English)**, **ita (Italian)**

Binarization

- □ Filter — Kind ADAPTIVE
- Coeff for Mean 0.7
- Coeff for StdDev 0.9

Scaling specifications

- □ Barline height — four
- □ Beam thickness — 0

Staves

- ☑ 1-line percussion staves ☑
- □ 4-line bass tablatures □
- ☑ 5-line standard staves □
- ☑ 5-line unpitched percussion staves ☑
- □ 6-line guitar tablatures □

Items

- □ Small heads □
- □ Small beams □
- □ Cross note heads □
- □ Tremolos □
- □ Fingering digits □
- □ Frets roman digits (I, II, IV...) □
- □ Plucking (p, i, m, a) □
- □ Partial whole rests □
- □ Multi-whole head chords □
- □ Chord names □
- ☑ Lyrics □
- □ Lyrics even located above staff □
- □ Articulations ☑

Processing

- □ Keep loaded gray images □
- □ Use of system indentation ☑
- □ Link augmentation dot to both shared heads □
- □ Implicit tuplets □

OK | Apply | Cancel

We select the **whole book** tab, not just the first sheet tab, because selections will need to apply to all sheets in book.

Here are the modifications made:

- We select a `Serif` font for better consistency with the input font.
- For OCR, `eng` and `ita` languages are all we need
- We select **both** 1-line percussion and 5-line unpitched percussion staves
- We don’t keep lyrics, since there are none in this score

> **NOTE**
> We don’t need to also select the “Cross note heads”, because we have selected the percussion staves. This “Cross note heads” feature is meant for cross heads on *standard* staves.

Customized drum-set

Since we are dealing with drum notation, we check the default drum-set.xml specifications. It is available in the Audiveris `res` (resources) folder. Its content is also displayed in this handbook section.

The default specifications should be OK for the 5-line staff (named Drumset in the input).

However, the 1-line staff in the input is named “Hand Clap”, so we will specify the hand clap “instrument” for an oval on mid-line (pitch-position 0).

In the default `drum-set.xml`, we have (excerpt):

```
    ...
  <staff line-count="1">
    <entry pitch-position="0"  motif="oval"    sound="null"/> <!-- Just a place-holder -->
  </staff>
    ...
```

Which basically means that no instrument is defined for a 1-line percussion staff.

So, we create a “personal” `drum-set.xml` that provides one overriding definition:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- My personal drum-set.xml-->
<drum-set>
  <staff line-count="1">
    <!-- Specific definition for Hand_Clap on the 1-line staves -->
    <entry pitch-position="0" motif="oval"  sound="Hand_Clap"/>
  </staff>
```

```
</drum-set>
```

And we put this file in the user Audiveris `config` folder.
On my Windows PC, the path to this folder is precisely:

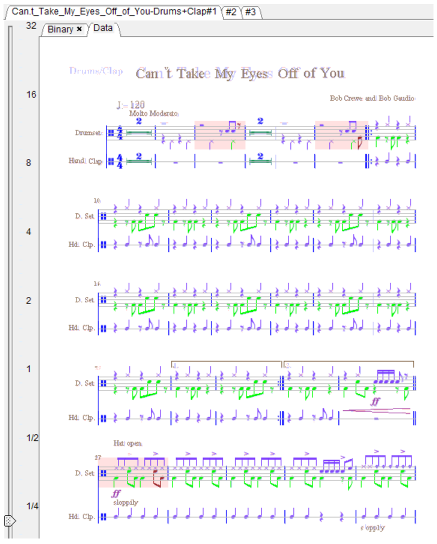
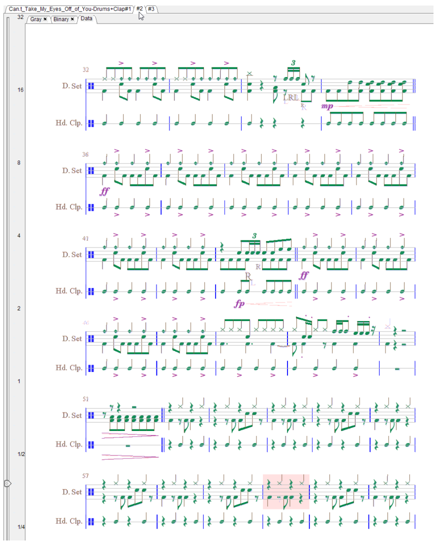
> C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\config

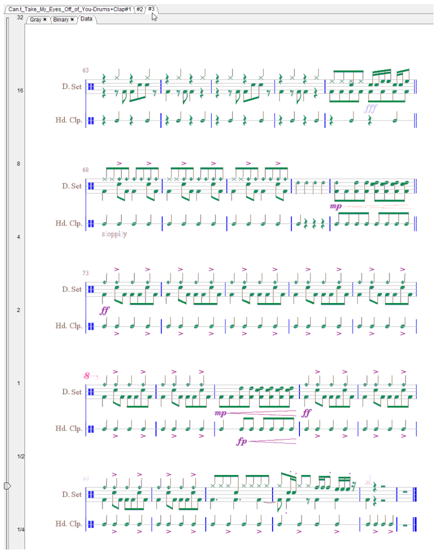
And that’s it. Audiveris will pick up this personal `drum-set.xml` file and load it to complete and/or replace the definitions provided by the default `drum-set.xml` file.

Raw results

We launch the whole book transcription, for example via `Book → Transcribe book`.

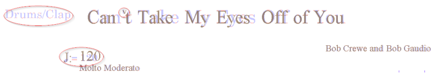
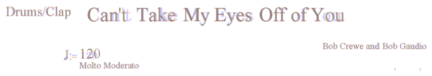
Two minutes later, we get:

Sheet ID	Raw result
Sheet#1	 Can.t_Take_My_Eyes_Off_of_You-Drums+Clap#1 #2 #3 Binary Data Drums/Clap Can't Take My Eyes Off of You Bob Crewe and Bob Gaudio ♩= 120 Molto Moderato Drumset Hand Clap D. Set Hd. Clp. Hat open ff ff
Sheet#2	 Can.t_Take_My_Eyes_Off_of_You-Drums+Clap#1 #2 #3 Gray Binary Data D. Set Hd. Clp. mp ff ff fp

Sheet ID	Raw result
Sheet#3	 Can't_Take_My_Eyes_Off_of_You-Drums-Clap#1 #2 #3 Gray Binary Data D. Set Hd. Clp.

Manual corrections

Sheet #1

	Header Text
Before	 Drums/Clap Can't Take My Eyes Off of You Bob Crewe and Bob Gaudio ♩= 120 Molto Moderato
After	 Drums/Clap Can't Take My Eyes Off of You Bob Crewe and Bob Gaudio ♩= 120 Molto Moderato

At the very beginning of sheet #1, we can see some header text partly recognized:

- “**Drums/Clap**”: Not recognized.
  We select the text with a lasso, then in the `Physicals` palette in the shape board we double-click on the `text` button. This manually calls the OCR on the selected text which is this time well recognized.
  The `Inter` board now presents the sentence “Drums/Clap” and proposes `CreatorLyricist` as its role. We manually change this role to `Number`.
- “**Can’t Take My Eyes Off of You**”: The apostrophe is mistaken for a separate “v” sentence.
  We select both sentences with a lasso and press the `Delete` key to remove them.
  With the underlying glyphs still selected, we double-click on the `text` button. The text is correctly OCR’ed as a single sentence and its role correctly set as

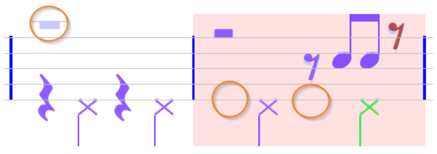
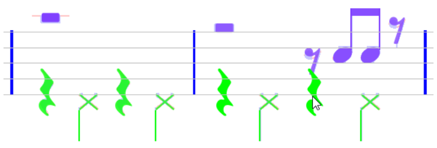
`Title`.

- ~~“� **= 120**”: Recognized as “J: 120”.~~
  ~~This is due to the quarter character (�) which the OCR does not handle correctly.~~
  ~~And there is currently no way to fix this.~~
- “**Molto Moderato**”: Correctly OCR’d and assigned the `Direction` role.
- “**Bob Crewe and Bob Gaudio**”: Correctly OCR’d and assigned the `CreatorComposer` role.

> UPDATE:
> With 5.4 release, the sentence “� **= 120**” is now correctly recognized as a metronome indication, meaning 120 quarters per minute.

Regarding rhythm, we have 3 measures displayed with a pink background. Their (local) ids are measures 4 and 8 in the first system and measure 27 in the last system.

We use `View → Show score voices` so that each voice is displayed in a specific color.

	Measures 3 & 4
Before	
After	

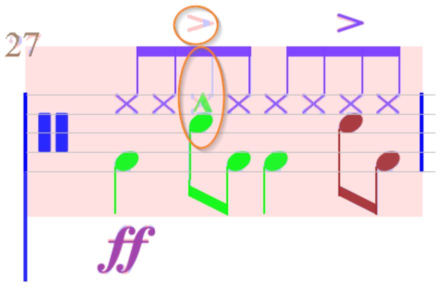
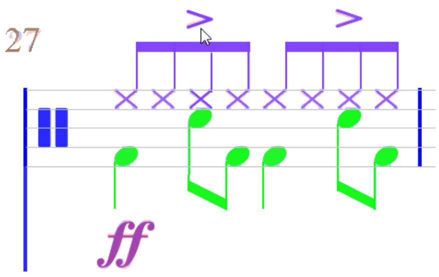
Measure 3: We observe a missing whole rest at the beginning of the measure.

- Detection failed because the whole rest is located out of the staff height.
- We select the glyph and assign it the WHOLE_REST shape using the `Rests` palette in shape board.

Measure 4: we can see the input lacks 2 quarter rests. This is especially obvious when compared with measure 3.

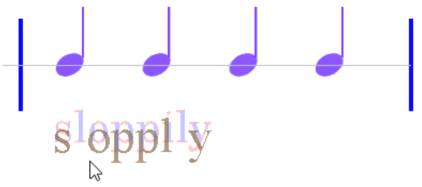
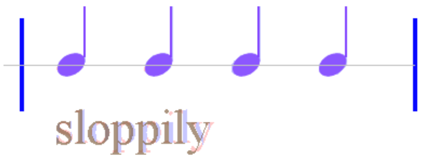
- So, we use a drag & drop to manually insert these missing quarter rests (this insertion triggers a RHYTHM re-processing for the containing measure which now shows a correct result in timing and voice mapping).

Measures 7 and 8: They are identical to measures 3 and 4, they need identical corrections.

	Measure 27
Before	
After	

Measure 27: A cross-head is missing, replaced by a strong accent.

- We delete the strong accent
- We select the stem glyph and assign it the stem shape
- We then insert a cross-head dragged from the `HeadsAndDot` palette in shape board.
- We also assign the accent just above this note (for example by selecting the underlying glyph, which triggers the glyph classifier; the `Accent` shape button appears in first place in the glyph classifier board; we simply press this `Accent` button).

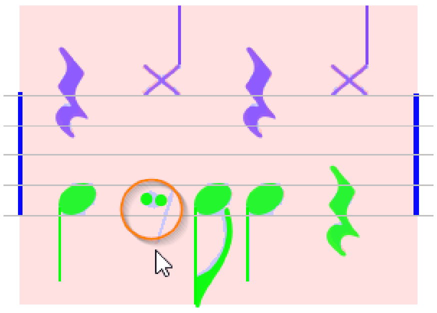
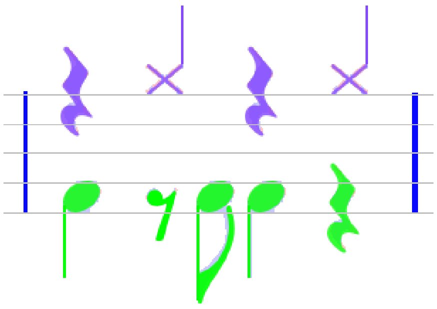
	Measure 31
Before	
After	

Measure 31: This last measure contains a direction word (“sloppily”) OCR’d as “s oppi y”.

- To fix this, the best way is to grab this item with a lasso, which selects the item as well as the underlying glyphs.
- We then delete the item
- With the underlying glyphs still selected, we manually run OCR via a double click on the "`text`" button in the `Physicals` set of the shape board.
- We then assign the sentence role as `Direction`.

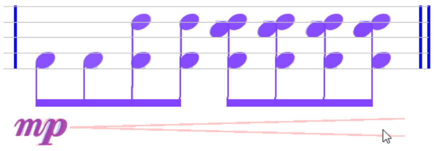
Sheet #2

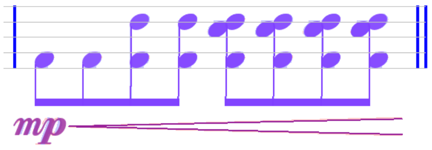
Just one measure (29) is displayed in pink.

	Measure 29
Before	
After	

This is due to an 8th rest mistaken for a pair of augmentation dots.

- With a lasso, we select the 8th rest underlying glyphs as well as the two dot items,
- We deassign the selected items,
- And click on the EIGHTH_REST button which appears at the top of the glyph classifier board.

	Measure 4
Before	

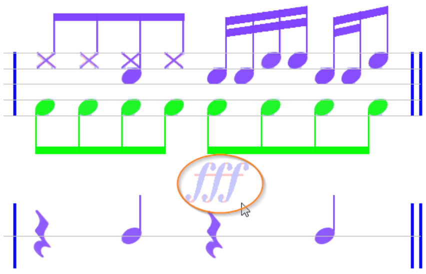
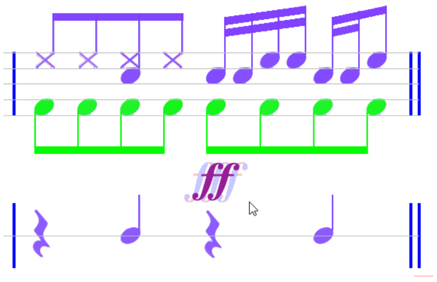
	Measure 4
After	

Measure 4: A missing crescendo wedge.

- We select the underlying glyph and use a double-click on the crescendo button in the `Dynamics` palette of the shape board.

A similar action fixes another missing crescendo below measure 12.

Sheet #3

	Measure 5
Before	
After	

Measure 5: A triple forte (fortississimo) is not recognized, because Audiveris doesn’t go beyond fortissimo. [1]

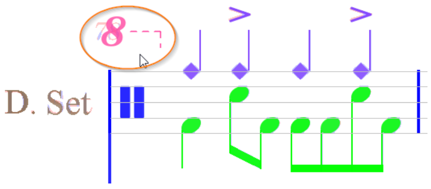
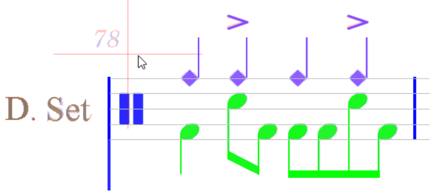
- So, we manually select just a fortissimo, for lack of a better choice.

Measure 6: a “sloppily” direction OCR’d as “s oppy y”.

- See similar action in sheet 1.

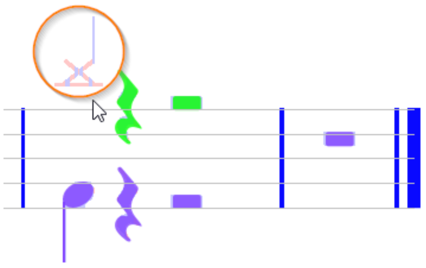
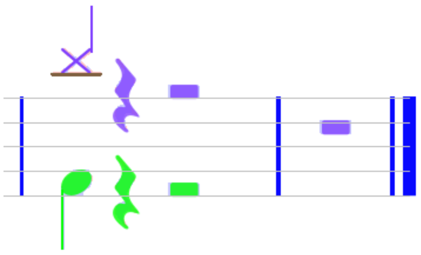
Measure 10: a missing crescendo sign.

- See similar action in sheet 2.

	Measure 16
Before	
After	

Measure 16: An “8” character is mistaken for the start of an octave shift.

- We simply delete it.

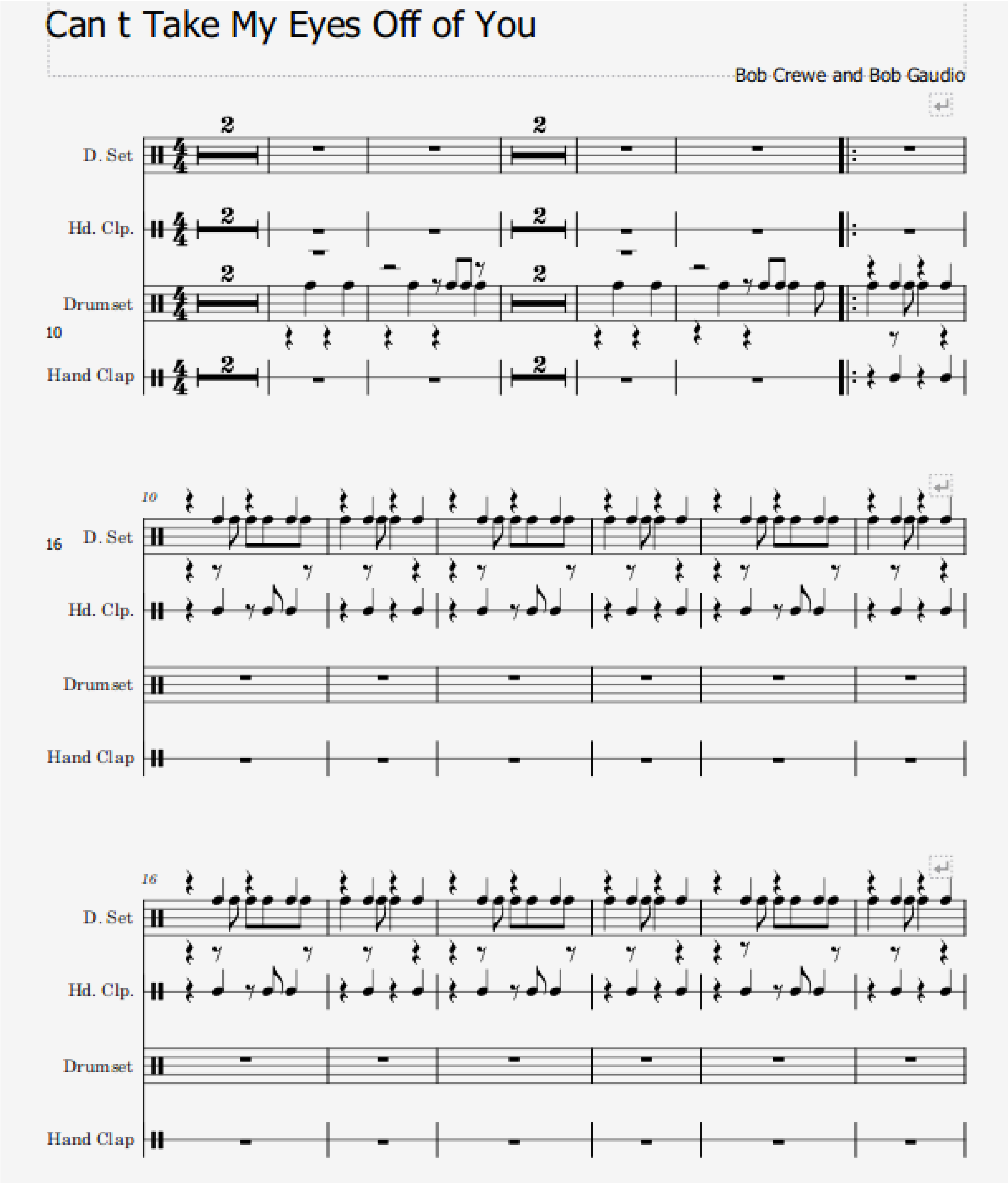
	Measure 25
Before	
After	

Measure 25: A missing quarter note.

- We select the stem glyph and assign it via a double-click on the stem button in the “`Physicals`” palette in the shape board.
- Then we drag a cross-head from the “`HeadsAndDot`” palette in the shape board into the proper location (the needed ledger line gets created automatically)

Logical parts

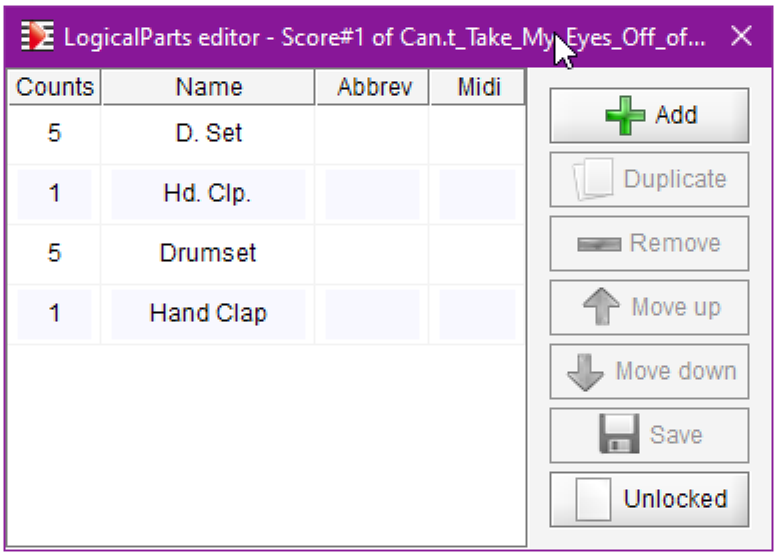
If we naively export our work to say MuseScore, we get something that starts like:



That is, for the OMR engine we have 4 separate logical parts:

- D. Set
- Hd. Clp.
- Drumset
- Hand Clap

And indeed, if within Audiveris we open the logical parts editor, we get this dialog:



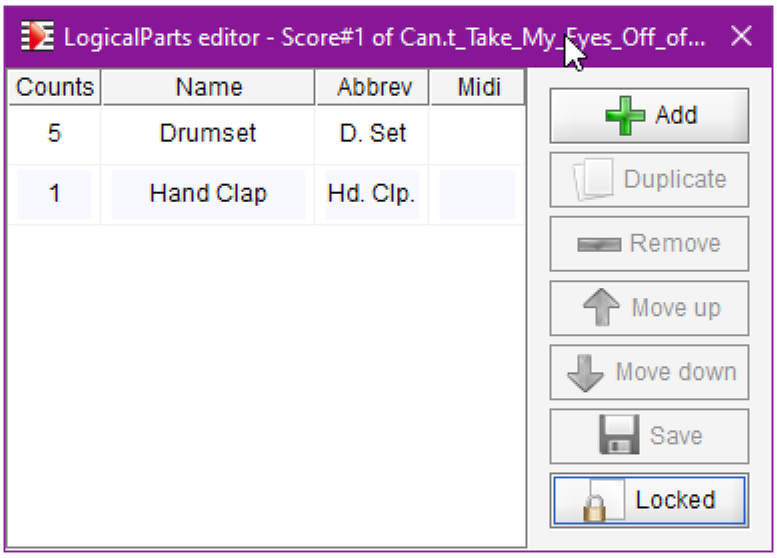
LogicalParts editor - Score#1 of Can.t_Take_My_Eyes_Off_of...

Counts	Name	Abbrev	Midi
5	D. Set		
1	Hd. Clp.		
5	Drumset		
1	Hand Clap		

Due to the OMR engine’s limited capabilities, there is no way to detect that “D. Set” and “Drumset” refer to the same logical part.

Let’s do this manually:

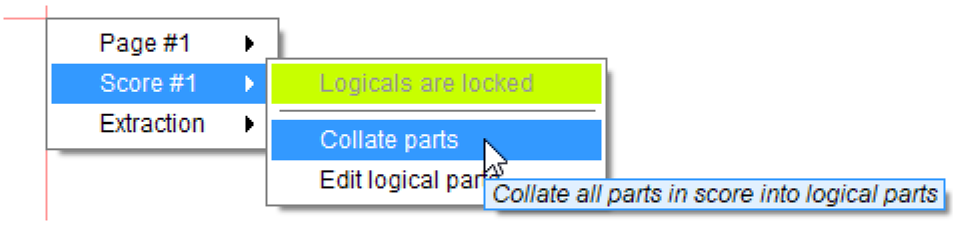
1. We select the first logical (D. Set) and copy its name
2. We then select the third logical (Drumset) and paste into the Abbrev field the copied name
3. We do the same between (Hd. Clp.) logical and (Hand Clap) logical
4. We can now remove the first 2 logicals (D. Set and Hd. Clp.)
5. We save (and thus lock) the logicals configuration



LogicalParts editor - Score#1 of Can.t_Take_My_Eyes_Off_of...

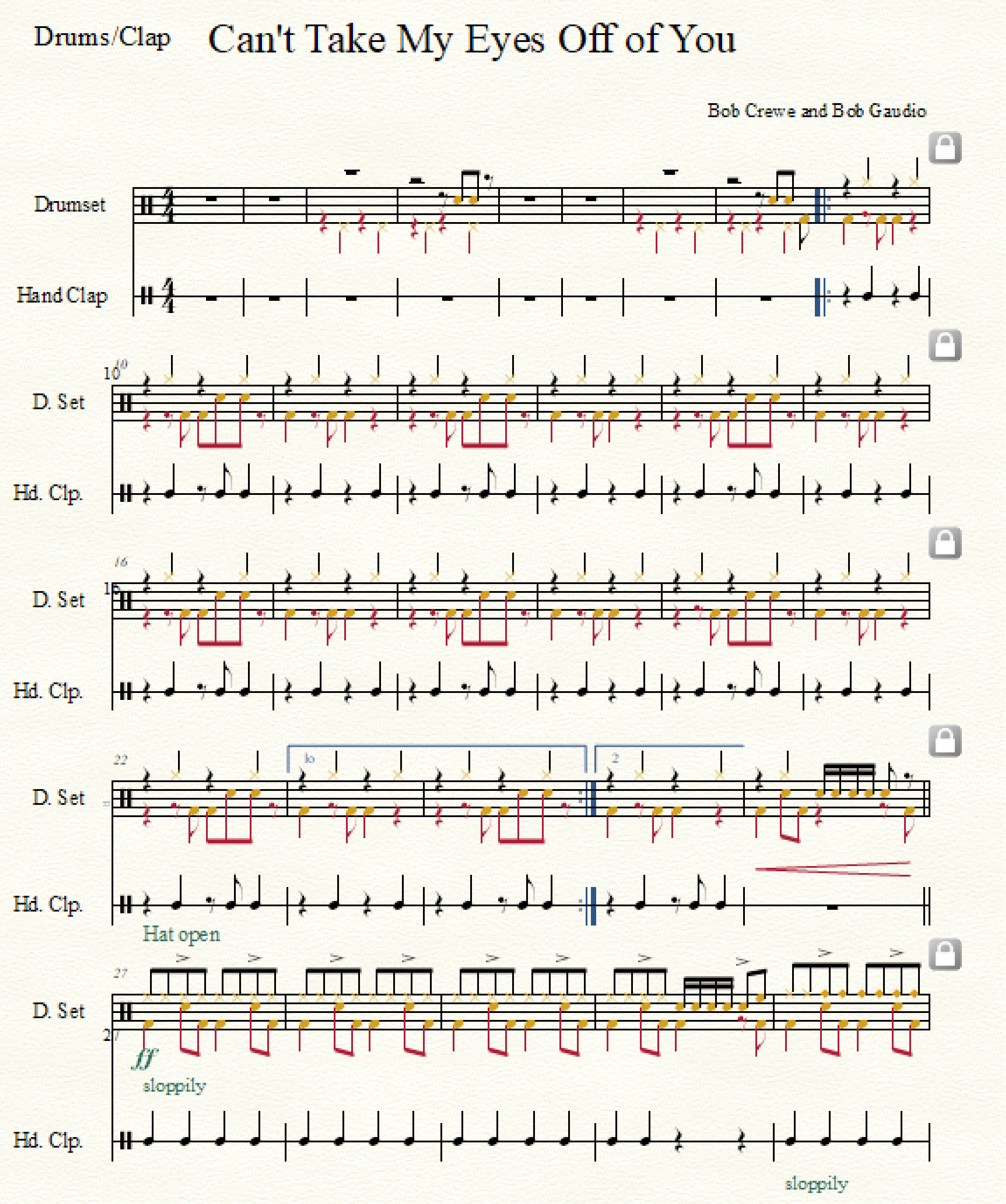
Counts	Name	Abbrev	Midi
5	Drumset	D. Set	
1	Hand Clap	Hd. Clp.	

We can finally apply the new configuration, with the logicals now locked:



No mapping warning is issued.

Export to MuseScore or Finale is now OK.
The first page in Finale looks like:



Footnotes

[1] We will need to add the fortississimo shape to Audiveris and retrain its glyph neural network with enough representative samples. ↩

Moonlight sonata

This famous *Sonata No. 14 “Moonlight”*, 1st movement, is a typical example of a score with implicit tuplets. The OMR engine is able to detect most of them but not all, so some user guiding actions are needed.

Input file available here

It is one of the numerous free scores (190 000+ as of this writing) available on the free-scores.com site.

Main score specificities:

- Book of 5 sheets
- Lots of implicit tuplets

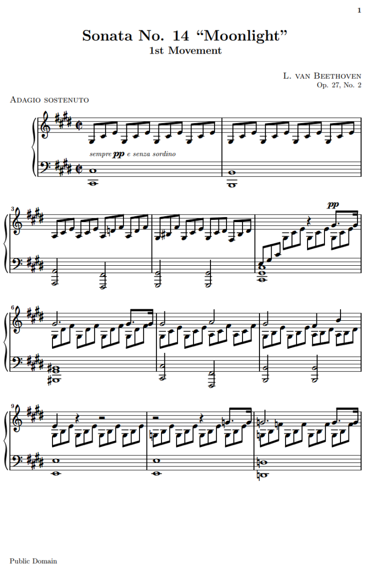
Main UI actions:

- Use of the `Chords` popup menu to fix voices, notably `Next in voice`, `Separate voices` and `Preferred voice` actions
- Manual insertion of tuplets

TABLE OF CONTENTS

Input

Here is the first image of a total of 5:

Sheet ID	Input image
Sheet#1	 1 Sonata No. 14 “Moonlight” 1st Movement L. van Beethoven Op. 27, No. 2 Adagio sostenuto sempre pp e senza sordino pp Public Domain

Book parameters

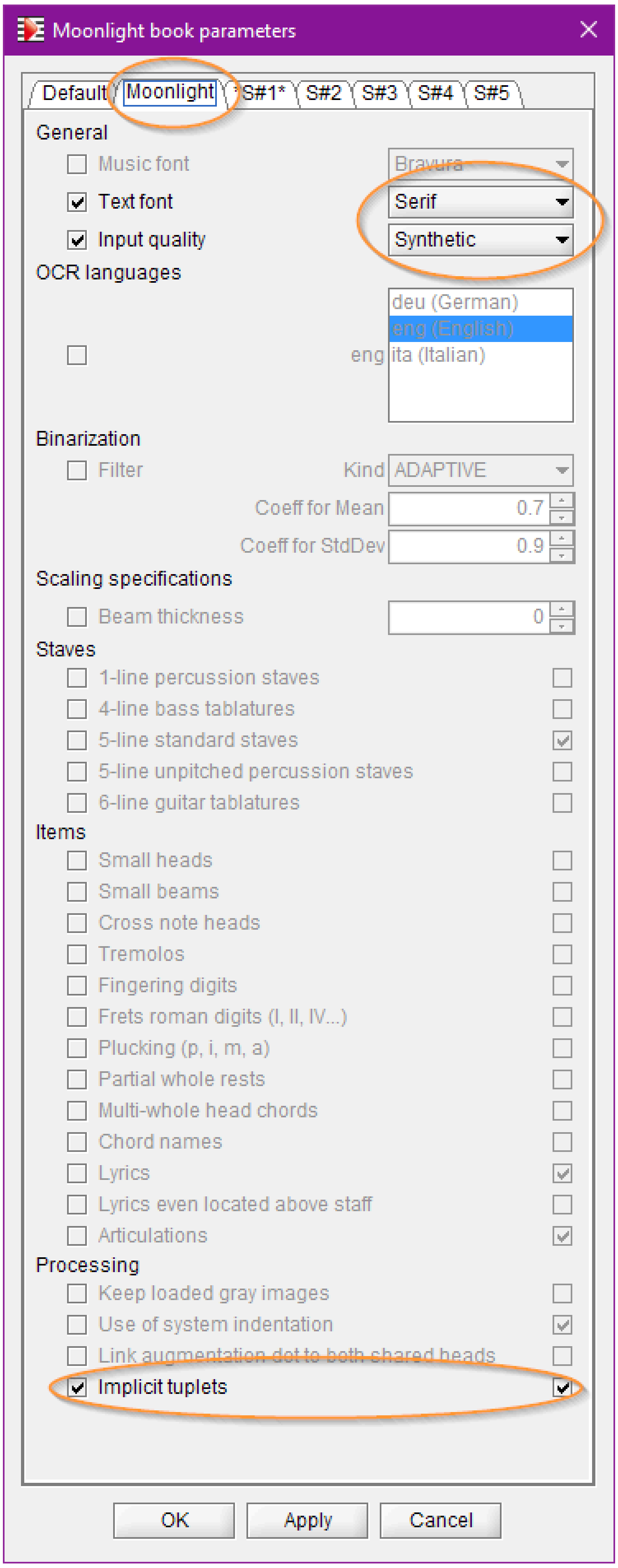
The input score has a “common cut” time signature (2/2), while each measure is made up of a sequence of 4 linked triplets. But none of these triplets is explicitly marked with a “3” tuplet sign.

So we’ll have to rely on the OMR engine to detect which groups of notes should be considered *triplets*.

Before launching the transcription, we have to set a few things in the book parameters:

- First, the settings should apply to the entire book, and not only to the (current) first sheet. So, we make sure to select the “Moonlight” book tab rather than the “S#1” sheet tab.

- The `Text font` is set to `Serif` rather than the default `Sans Serif` font. This has no impact on the OCR efficiency, the effect is mainly a more faithful display of texts.
- The `Input quality` is set to `Synthetic` rather than the default `Standard`. This higher quality results in a more strict checking and thus less mistakes.
- Finally, setting the `Implicit tuplets` to ON is absolutely mandatory!



General

☐ Music font Bravura

☑ Text font Serif

☑ Input quality Synthetic

OCR languages

☐ eng deu (German) eng (English) ita (Italian)

Binarization

☐ Filter Kind ADAPTIVE

Coeff for Mean 0.7

Coeff for StdDev 0.9

Scaling specifications

☐ Beam thickness 0

Staves

☐ 1-line percussion staves ☐

☐ 4-line bass tablatures ☐

☐ 5-line standard staves ☑

☐ 5-line unpitched percussion staves ☐

☐ 6-line guitar tablatures ☐

Items

☐ Small heads ☐

☐ Small beams ☐

☐ Cross note heads ☐

☐ Tremolos ☐

☐ Fingering digits ☐

☐ Frets roman digits (I, II, IV...) ☐

☐ Plucking (p, i, m, a) ☐

☐ Partial whole rests ☐

☐ Multi-whole head chords ☐

☐ Chord names ☐

☐ Lyrics ☑

☐ Lyrics even located above staff ☐

☐ Articulations ☑

Processing

☐ Keep loaded gray images ☐

☐ Use of system indentation ☑

☐ Link augmentation dot to both shared heads ☐

☑ Implicit tuplets ☑

We can save the book right now, to save the book parameters with it.

Raw results

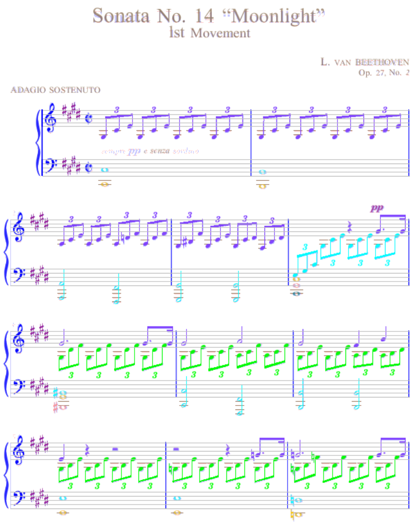
We can launch transcription of the entire book, for example via the `Book → Transcribe book` pull-down menu.

With Audiveris 5.4, the transcription took exactly 2 minutes for 5 sheets on a 14-year old laptop running Windows 10.

We use the `View → Show score voices` menu item to display voices with different colors, and the `View → Show chord IDs` item to display the chord numeric IDs.

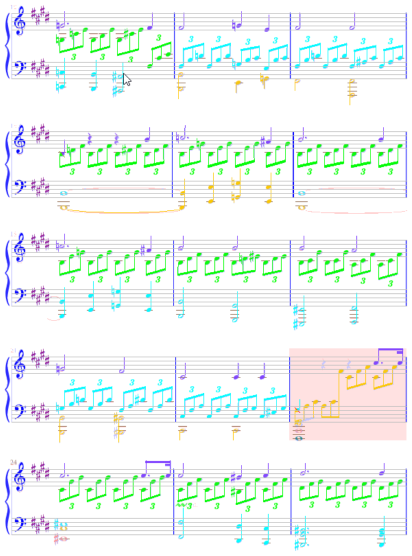
Sheet #1

No measures to fix.

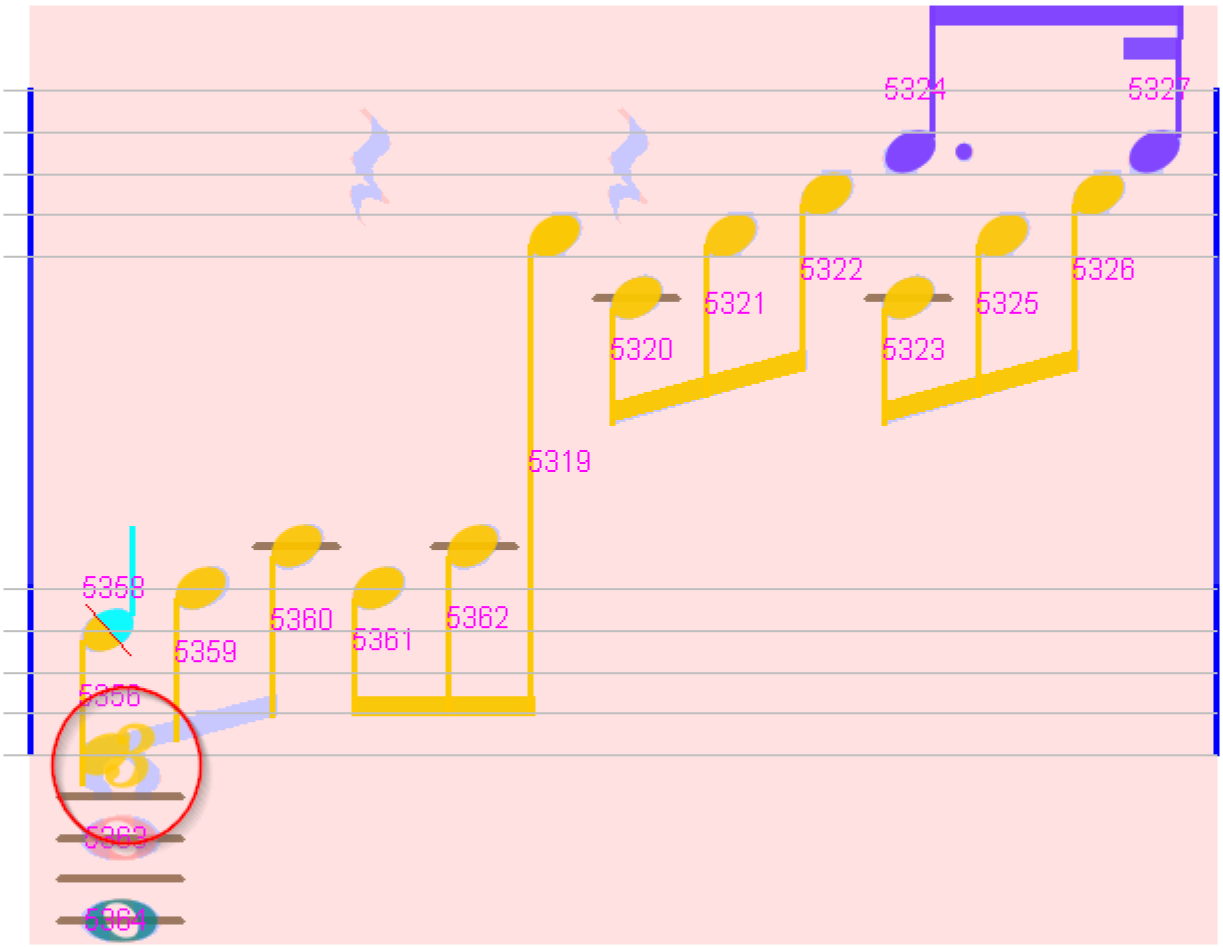
Sheet ID	Raw results
Sheet#1	 Sonata No. 14 "Moonlight" 1st Movement L. VAN BEETHOVEN Op. 27, No. 2 ADAGIO SOSTENUTO pp

Sheet #2

One measure in pink

Sheet ID	Raw results
Sheet#2	

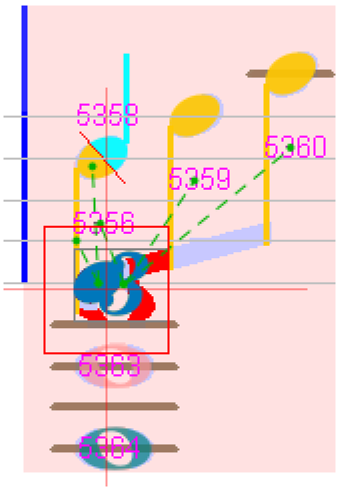
Measure #12



Here we have a mess on the lower left side:

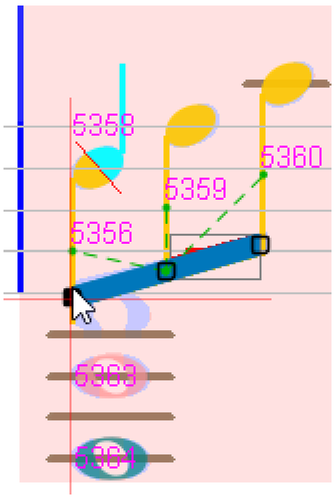
- A non-recognized beam
- A false note head
- A tuplet sign

We first delete the note head and the tuplet sign:

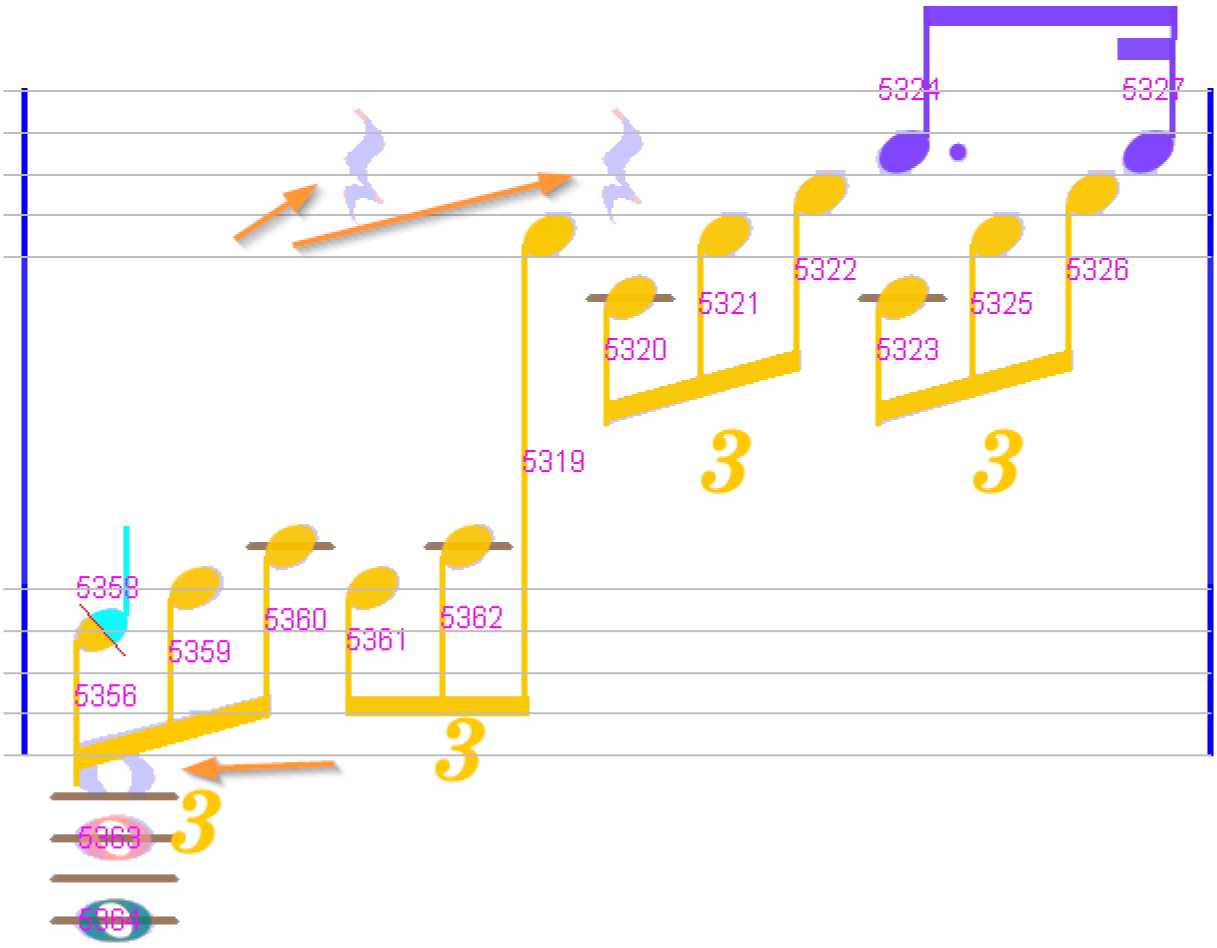


We then assign the Beam shape to the underlying glyph.

We enter the edit mode via a double click on the beam, to extend the beam to the left, until it embraces the left-most stem (at this point the beam shows 3 links to the embraced 3 stems).



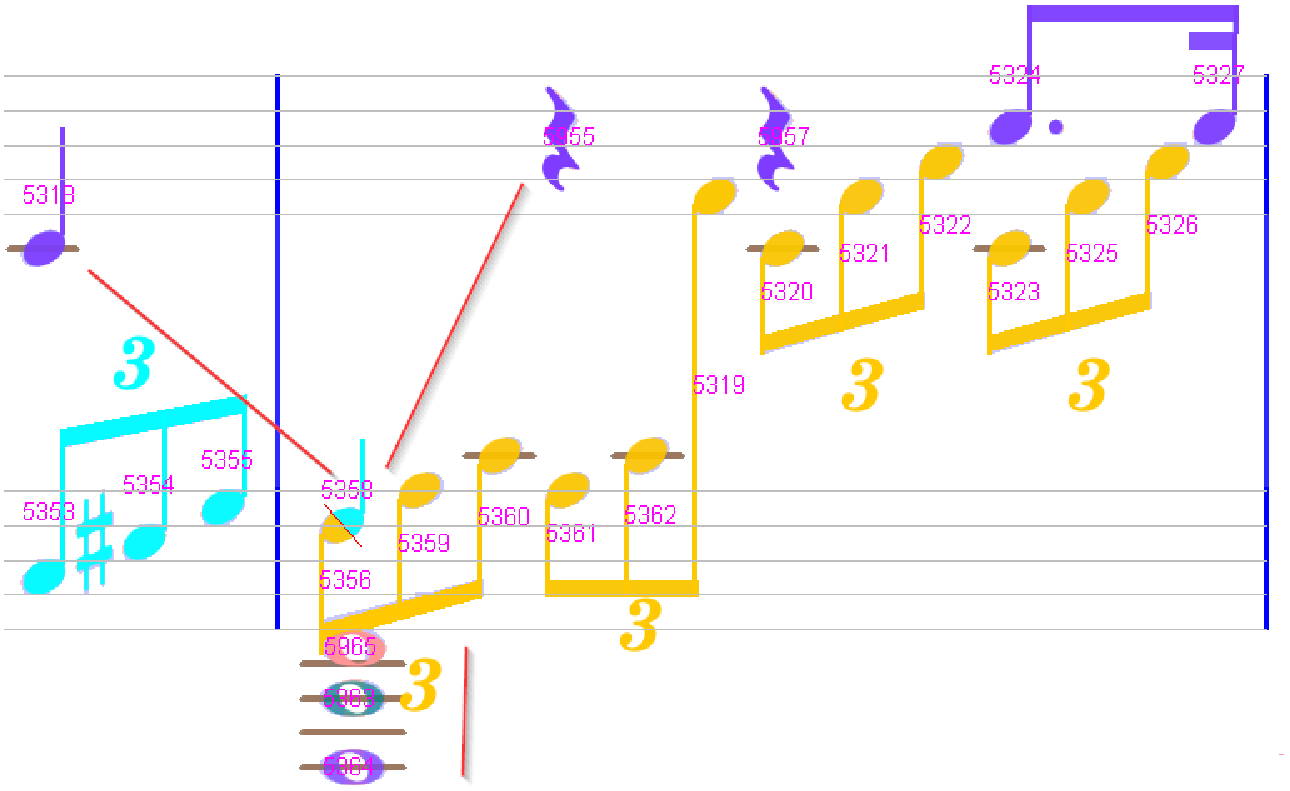
And the measure turns white.



However, the upper staff exhibits 2 unrecognized quarter rests and the lower staff a missing whole head.

We assign each of the quarter rests for example by clicking on the underlying glyph, which triggers the glyph classifier, and we simply click on the top 1 shape (quarter rest).

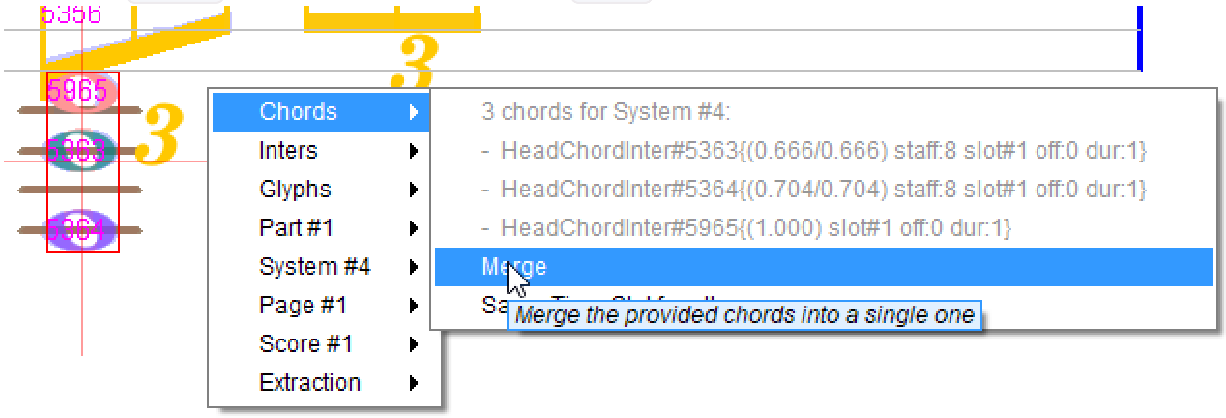
For the non-recognized whole head, we have no correct underlying glyph. So, we simply drag a whole head from the ShapeBoard and drop it on the proper location.



We could stop at this point, because it’s OK as far as the rhythm is concerned.

We can also further refine the result with two actions:

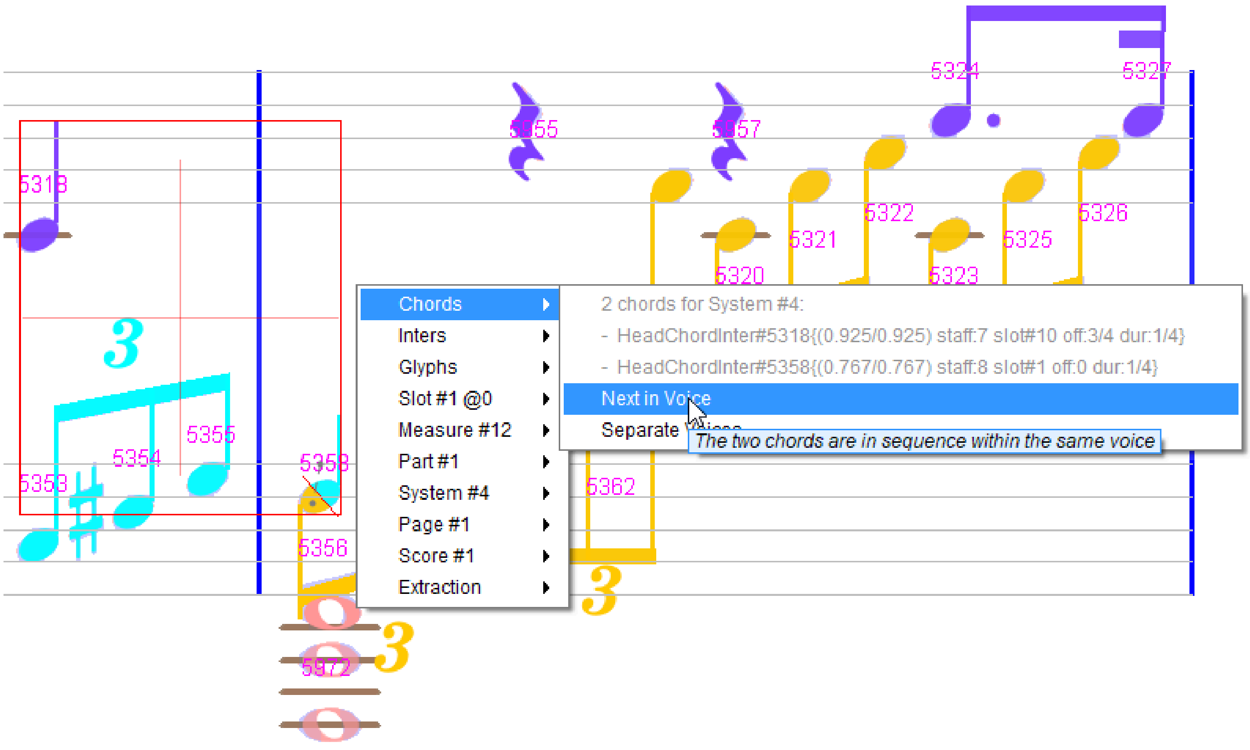
- At the bottom of the lower staff, we could decide to merge the 3 whole heads into a single chord. To do so, we select the 3 whole heads, then in the pop-up `Chords` menu, we select the `Merge` action.



3 chords for System #4:
- HeadChordInter#5363{(0.666/0.666) staff:8 slot#1 off:0 dur:1}
- HeadChordInter#5364{(0.704/0.704) staff:8 slot#1 off:0 dur:1}
- HeadChordInter#5965{(1.000) slot#1 off:0 dur:1}

Merge the provided chords into a single one

- Between upper and lower staff, we can consider that the quarter note (chord #5358 in voice 5 cyan) should rather belong to the voice 1 (blue). To do so, we use the `Next in voice` action twice:
  - Between the last quarter #5318 of previous measure and the chord #5358
  - Between chord #5358 and the first quarter rest #5955 in the upper staff.

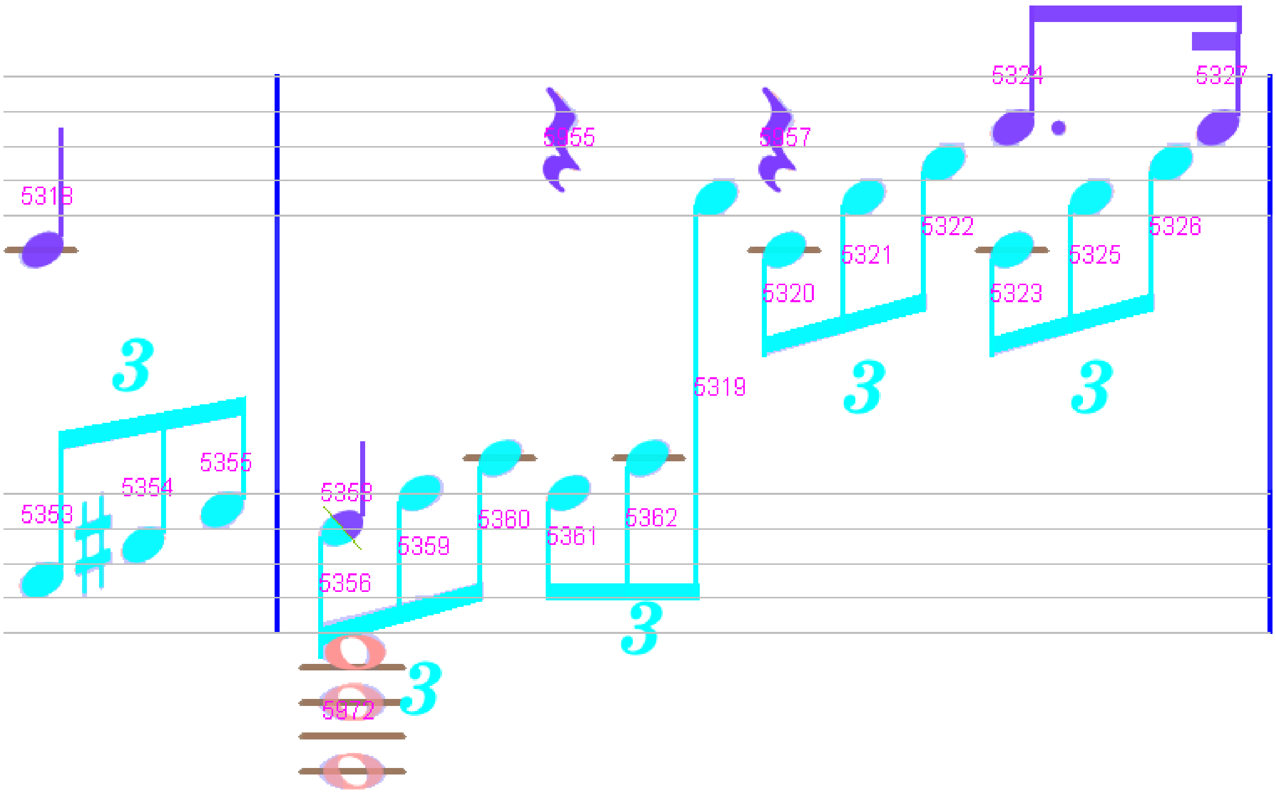


2 chords for System #4:

- HeadChordInter#5318{(0.925/0.925) staff:7 slot#10 off:3/4 dur:1/4}
- HeadChordInter#5358{(0.767/0.767) staff:8 slot#1 off:0 dur:1/4}

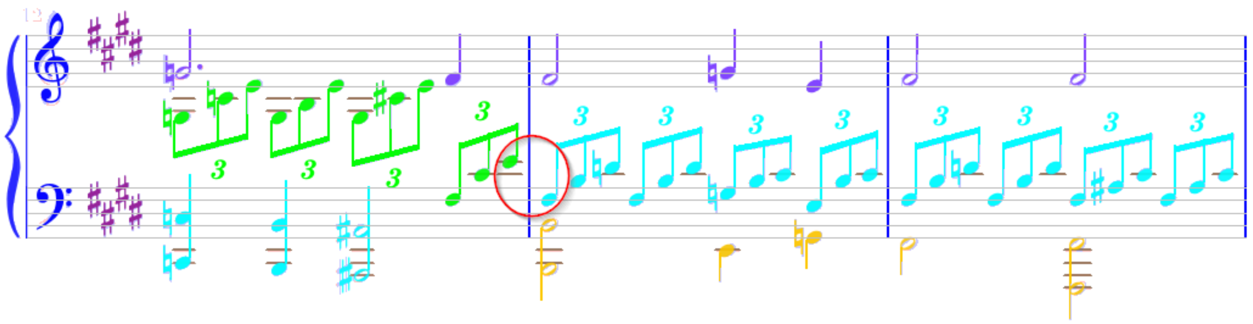
The two chords are in sequence within the same voice

Here is the final status of this measure:



Measures #1, #2, #3

Let’s have a look back at the top of this page, the system #1:



All measures in this system are displayed in white and would be correctly exported.

However, we can notice that the “intermediate” voice varies between the three measures of the system:

- Green (voice #2) in measure #1
- Cyan (voice #5) in measures #2 and #3

And similarly, the “lower” voice appears in cyan (voice #5) in measure #1 and in orange (voice #6) in the following two measures.

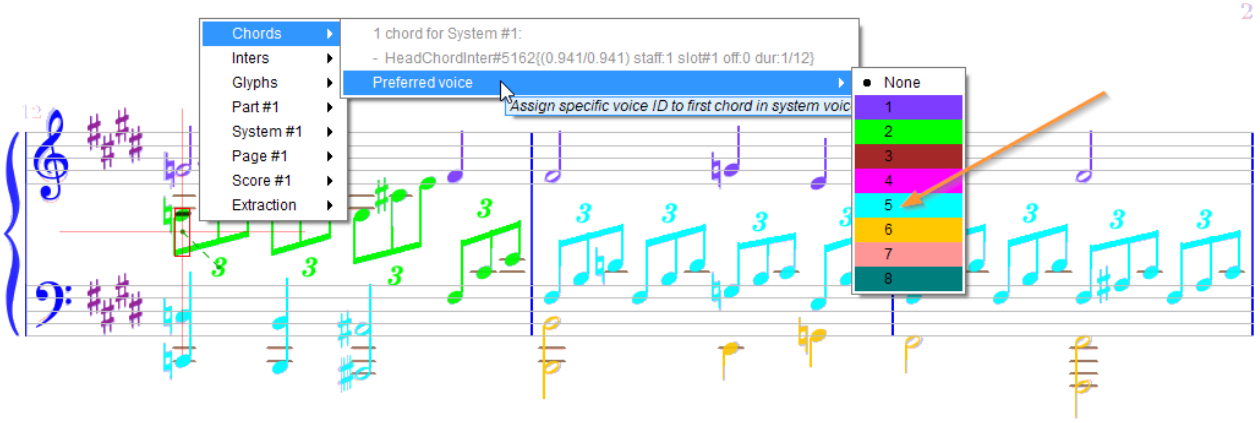
We may wish to further improve this system.

The current policy of the rhythm algorithm is to assign the voice number at the very beginnning of a measure according to:

- The containing staff (1-4 for staff #1, 5-8 for staff #2)
- The vertical rank within the staff.

We can override this policy via two ways:

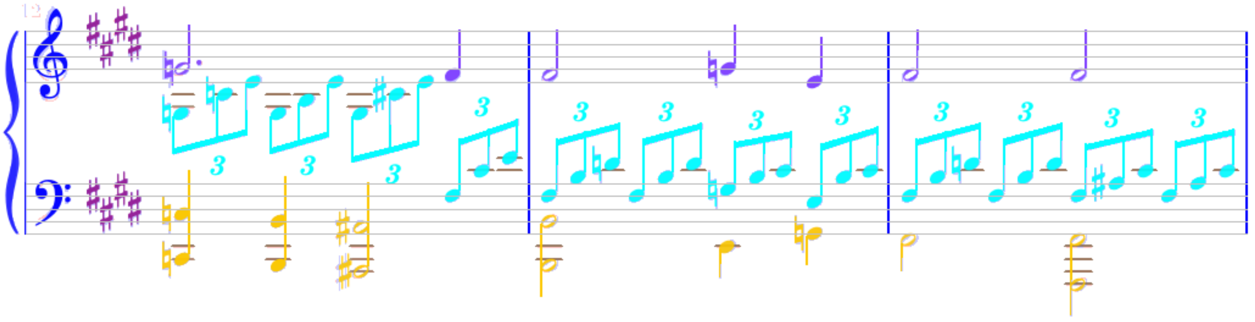
- Within the *same* system, we can use the `Next in voice` action between the last note of a measure and the first note of the following measure,
- At the *beginning* of a system, we can use the `Preferred voice` action on the note of a starting chord to assign an absolute voice number.


1 chord for System #1:

- HeadChordInter#5162{(0.941/0.941) staff:1 slot#1 off:0 dur:1/12}

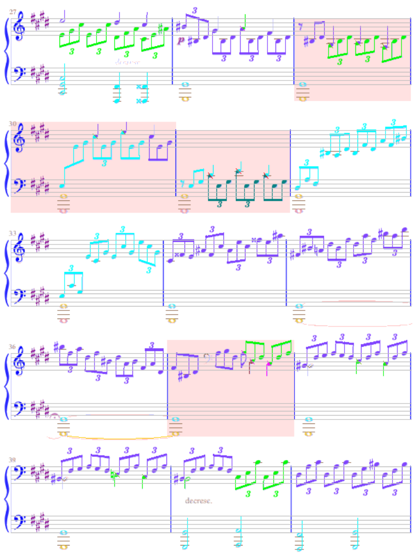
Assign specific voice ID to first chord in system voic

Doing so for the “intermediate” and for the “lower” voices, we get this final result:

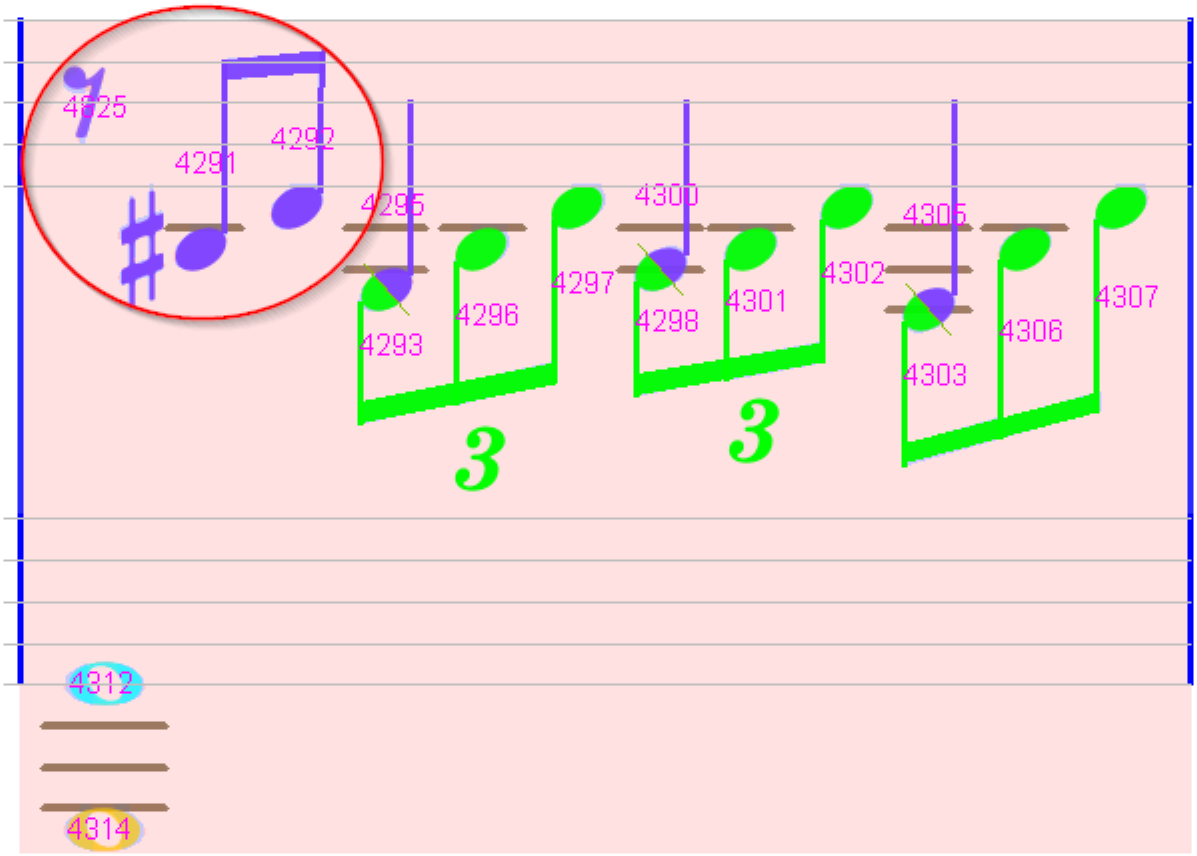


Sheet #3

We have four measures to fix.

Sheet ID	Raw results
Sheet#3	

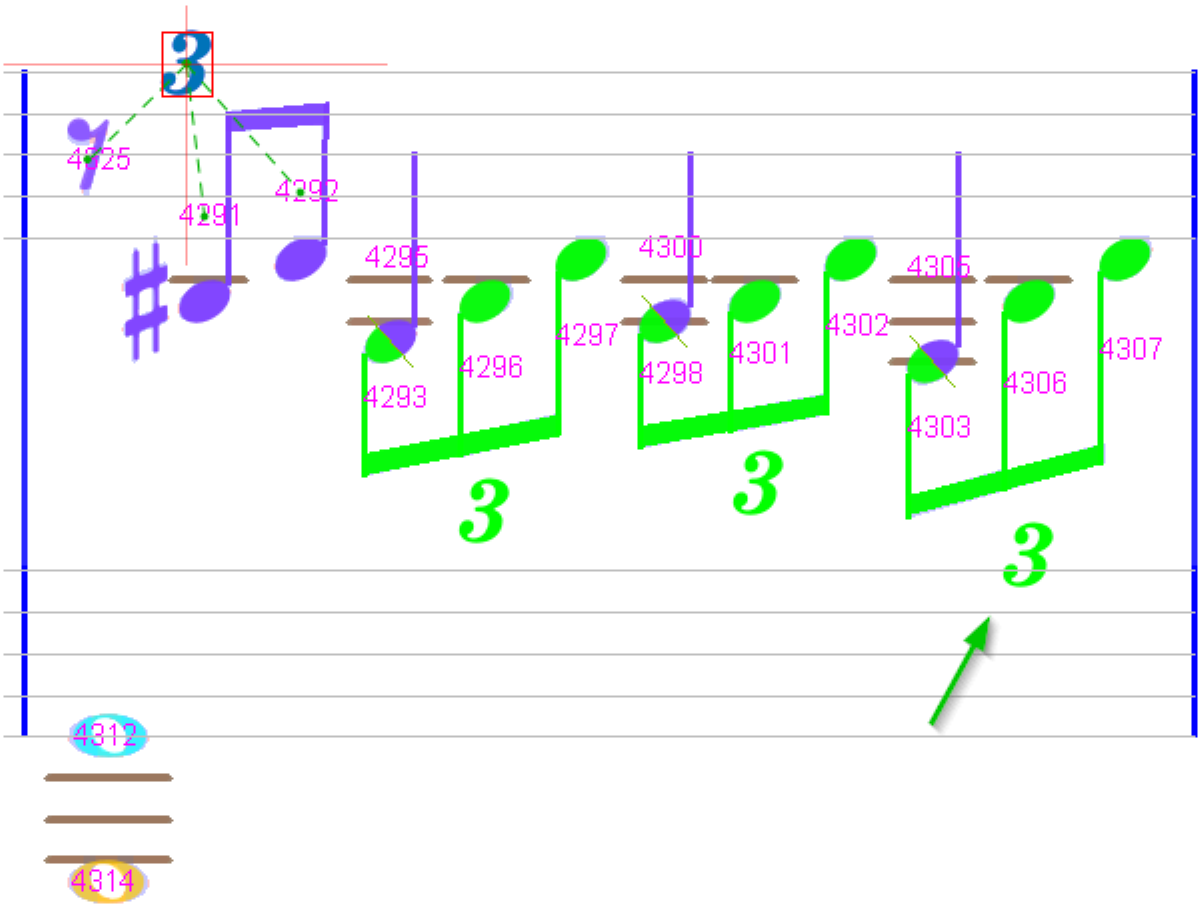
Measure #3



We can observe that the 3 first chords in the upper staff should have a tuplet.

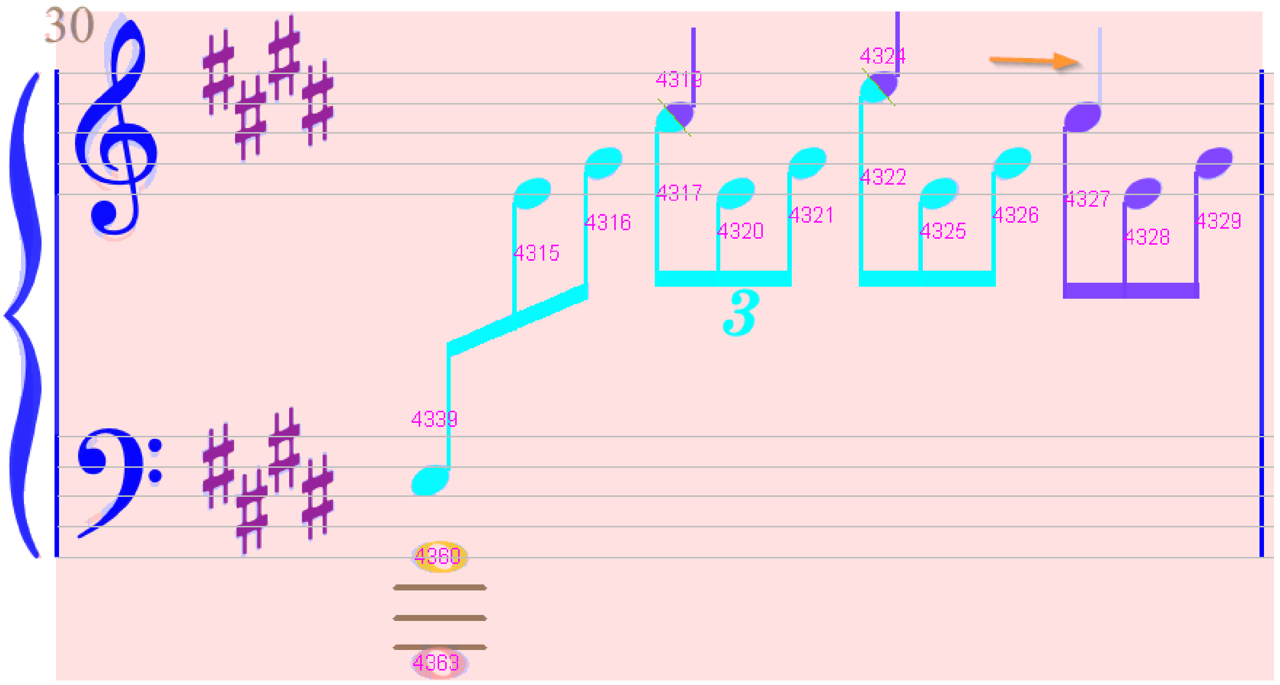
We manually insert a tuplet-3, dragged from the Shape board (the `BeamsEtc` subset) and dropped at proper location.

> While being dragged, a tuplet is displayed with its potential links with the embraced chords. These links, shown as green dashed lines, are a good visual aid to decide where to precisely drop the tuplet.



We can notice that this manual fix at the start of the measure resulted in the automatic insertion of the last implicit tuplet at the end of the measure.

Measure #4

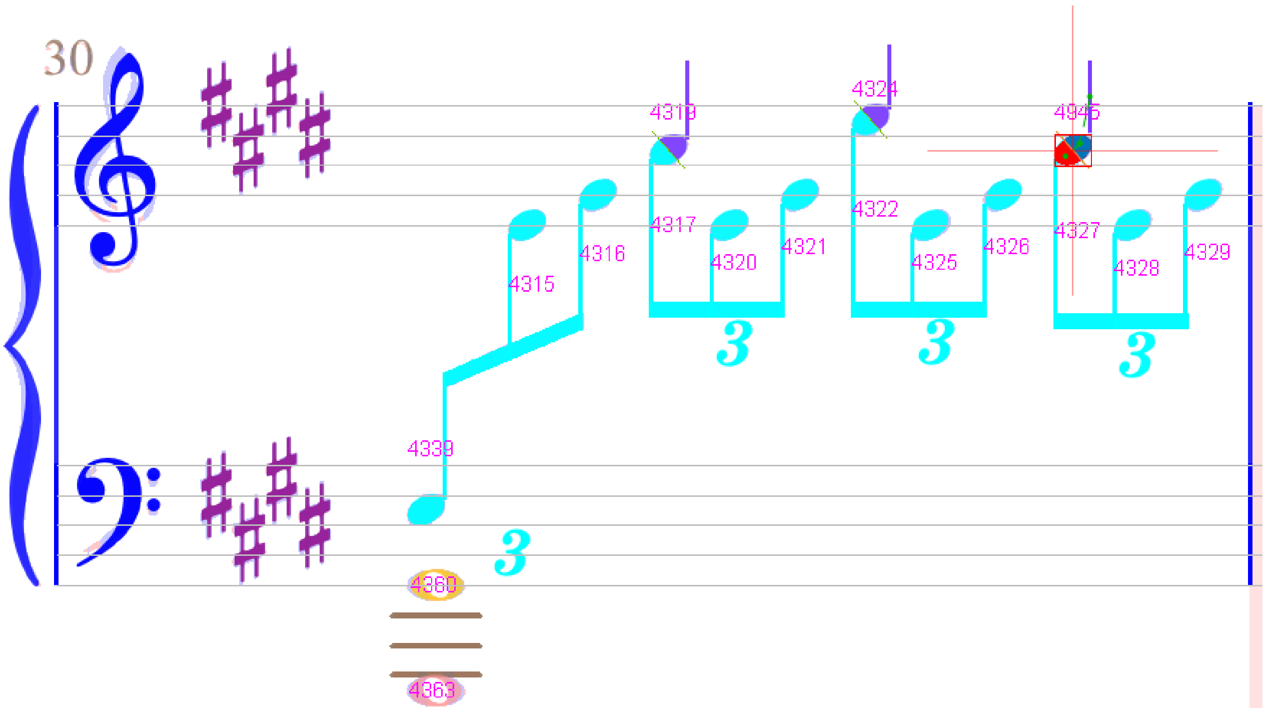


On the upper right, a stem has not been detected.

We select the underlying glyph and in the Shape board (Physicals set) we double-click on the stem button.

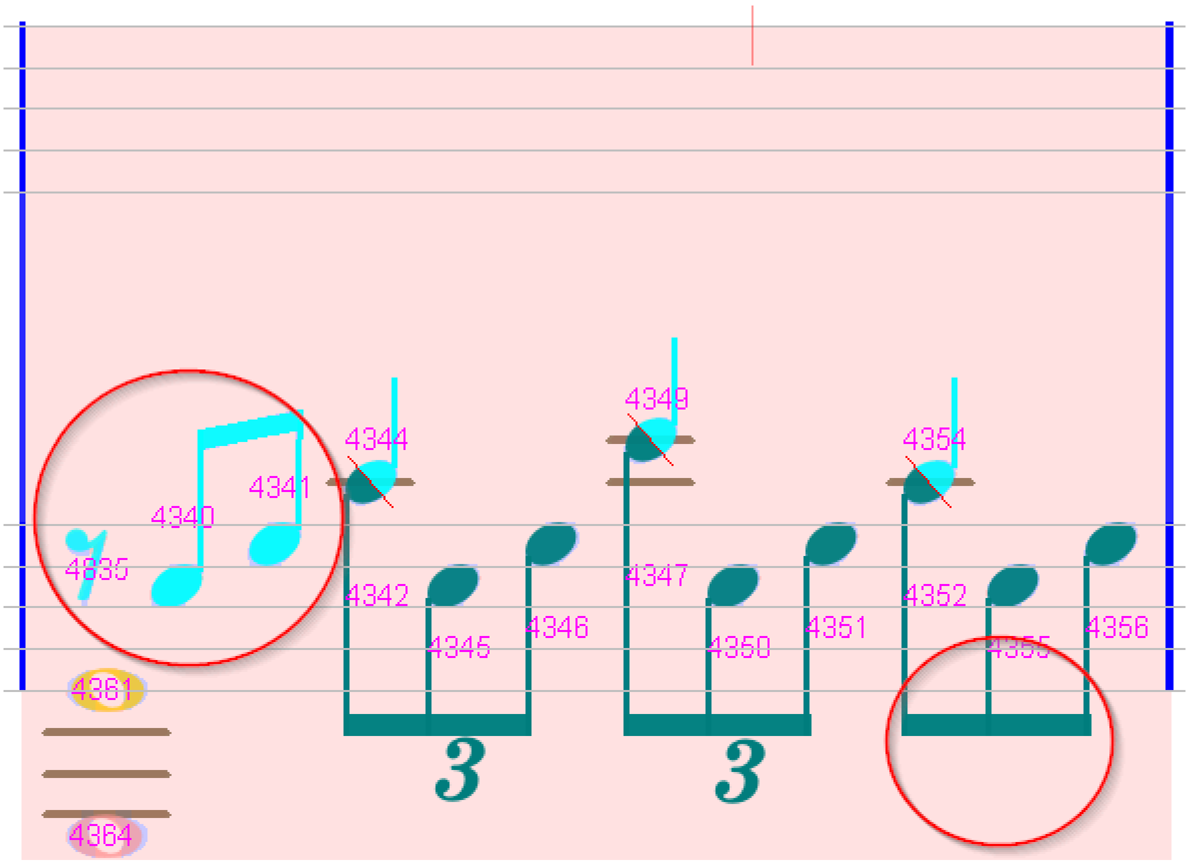
We now have a note head with two stems, one linked on the head left side, and the other (the new one) on the head right side, but not yet linked.

To link head and stem, we press the mouse on the stem, drag to the head and release the mouse.



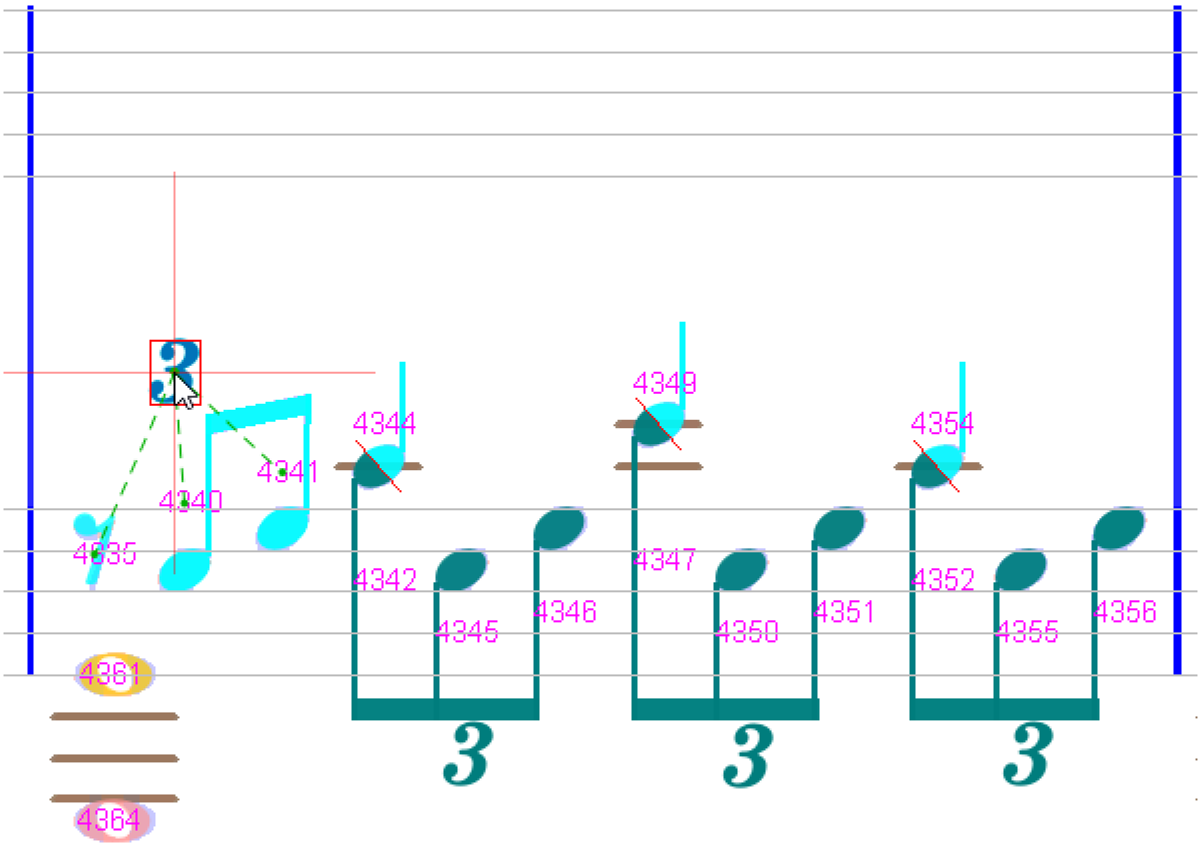
The physical note head is now split into 2 logical half-heads, and the measure turns white.

Measure #5

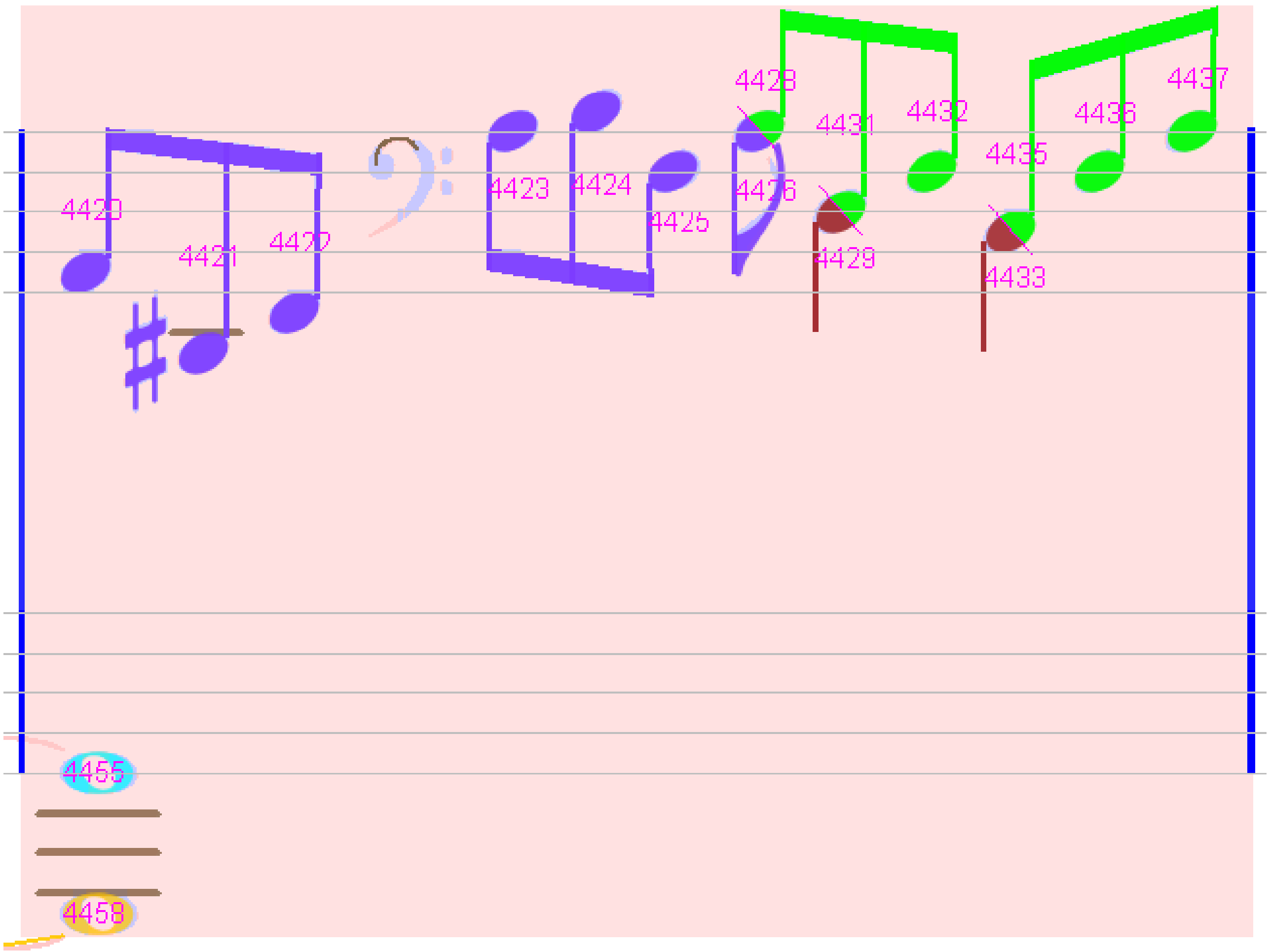


We have missing implicit tuplets on two locations.

So, we manually insert a tuplet of the left location (at the beginning of the measure). The second (implicit) tuplet is automatically inserted.



Measure #11

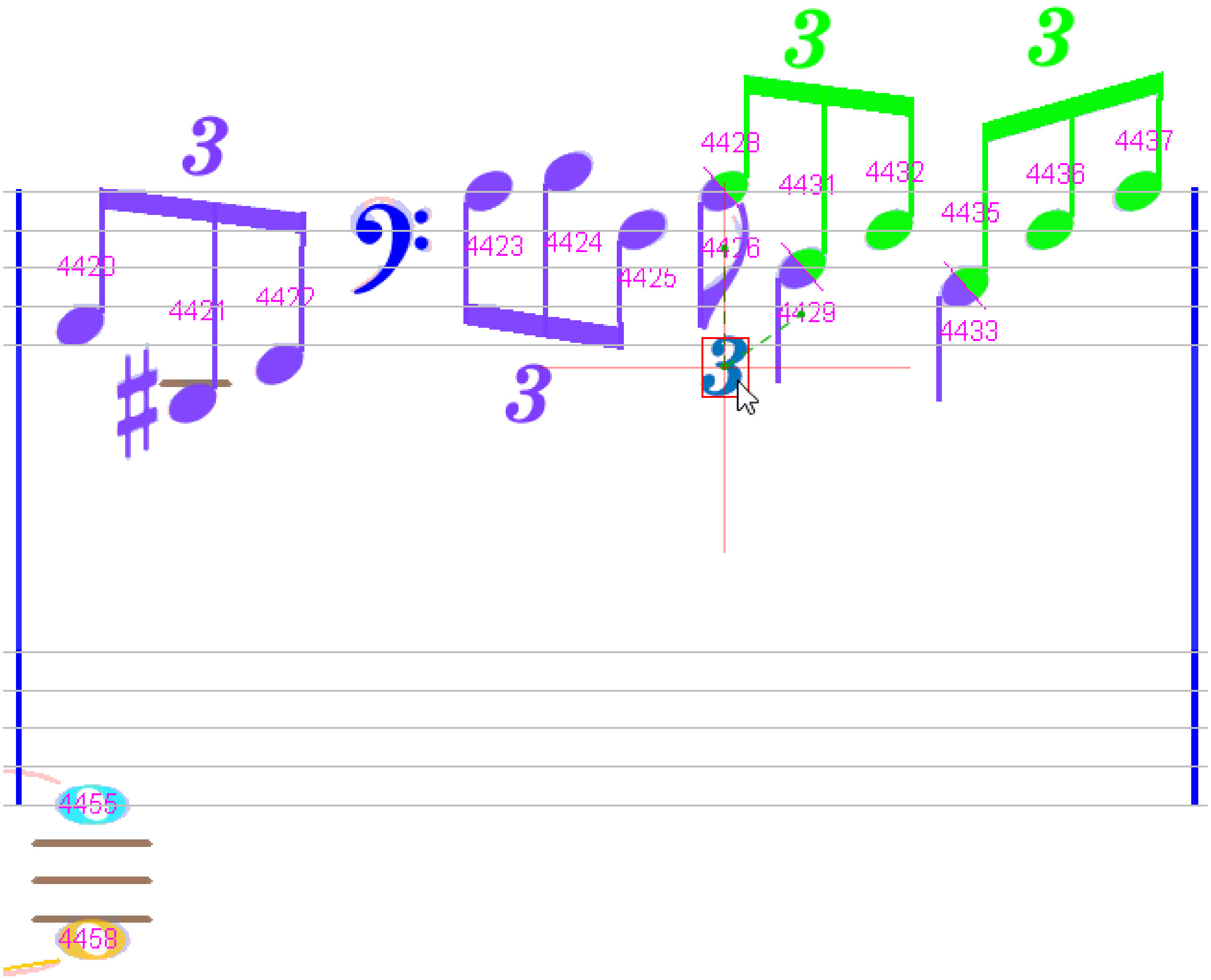


A false slur detected (to be manually deleted) and a bass clef not detected (to be manually assigned).

Regarding the tuplets, there is none, certainly due to the complexity at the end of the measure.

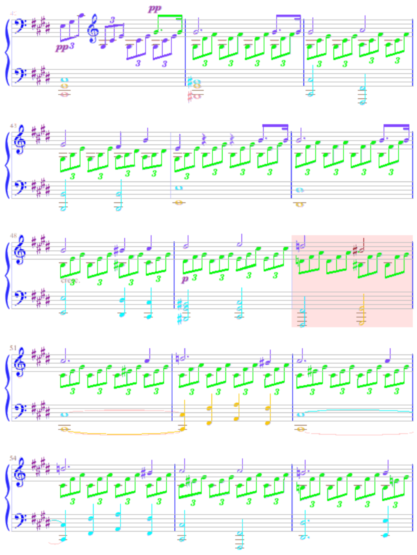
So we insert them manually from left to right.

When the 3rd one is inserted between the down eighth and the down quarter, the measure turns white with the upper tuplets automatically inserted:

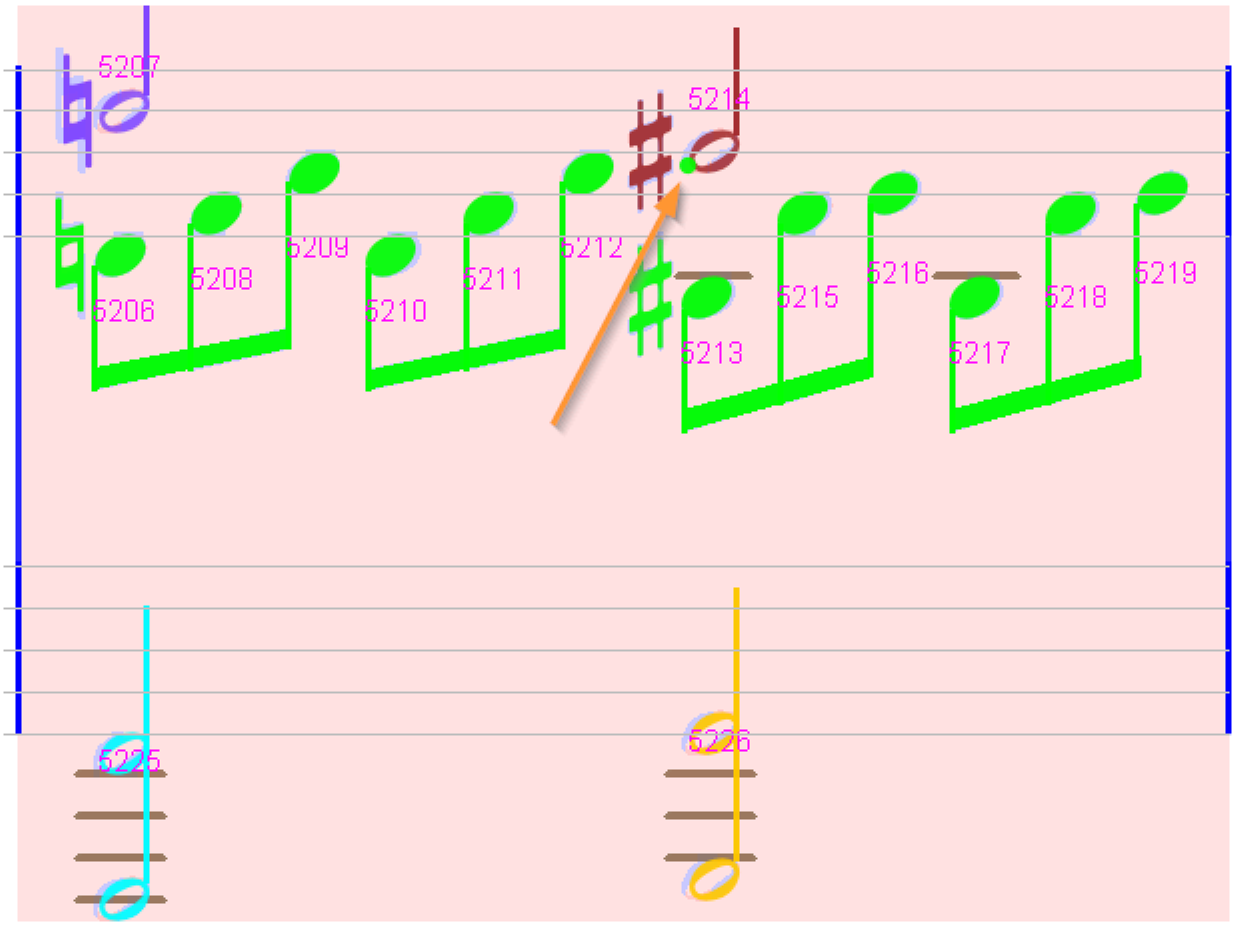


Sheet #4

Just one measure to fix.

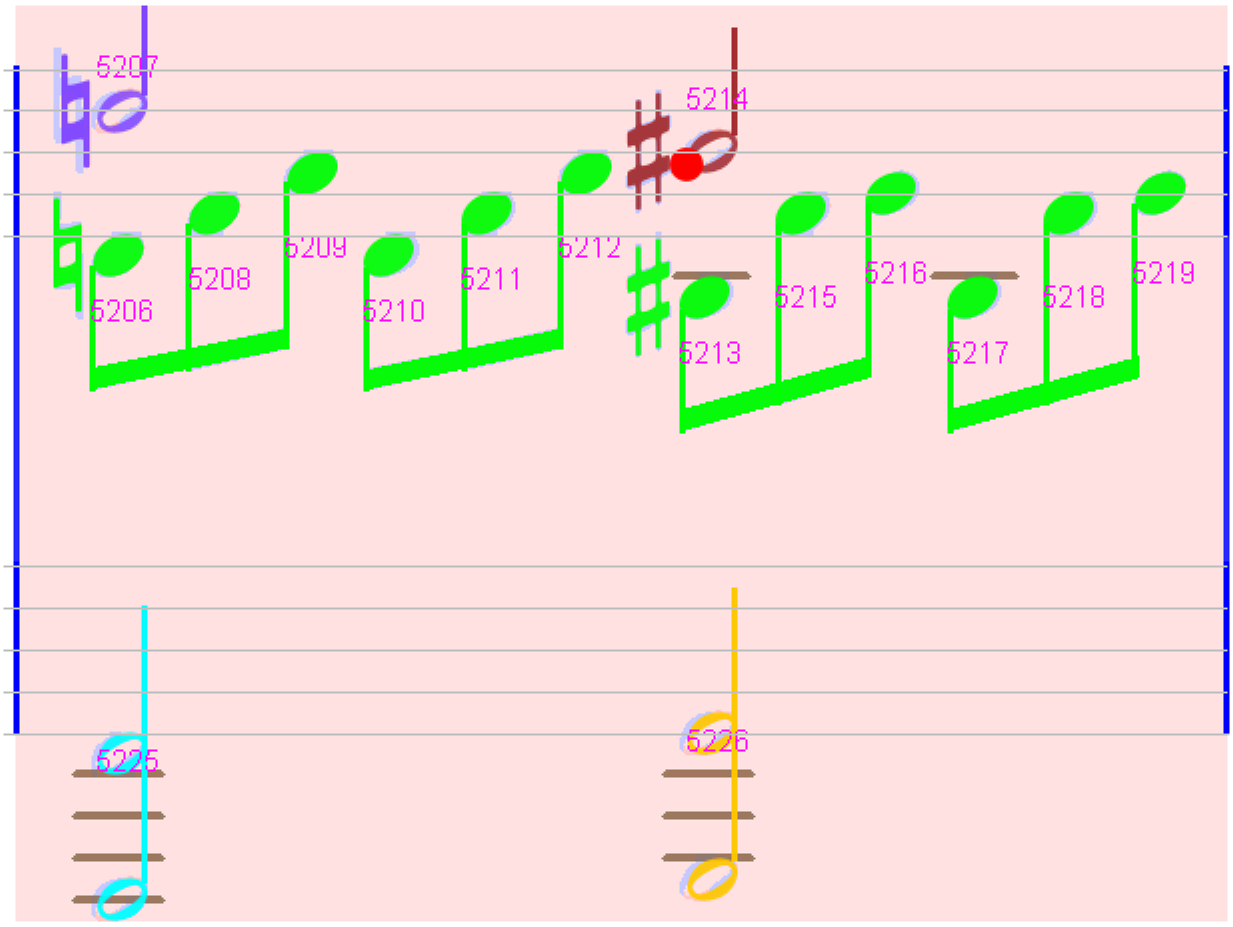
Sheet ID	Raw results
Sheet#4	

Measure #9

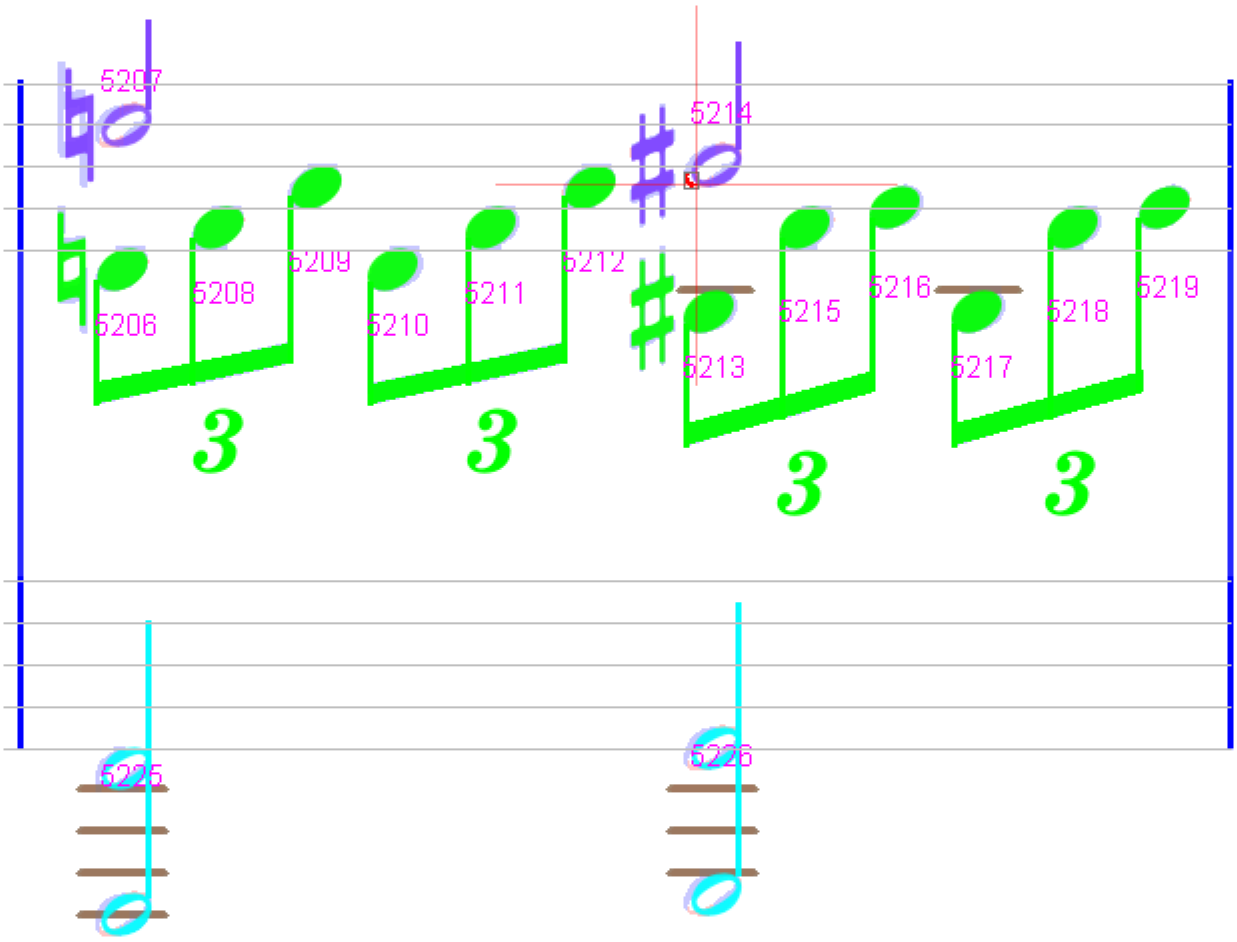


A small glyph has been mistaken for a dot, considered as an augmentation dot.

This kind of error is not often easy to detect visually. This is why we now have a view option, activated via the `View → Show jumbo inters` menu or the `F7` function key.
When set to ON, all the augmentation dots are highlighted as big red dots, as in the following picture:

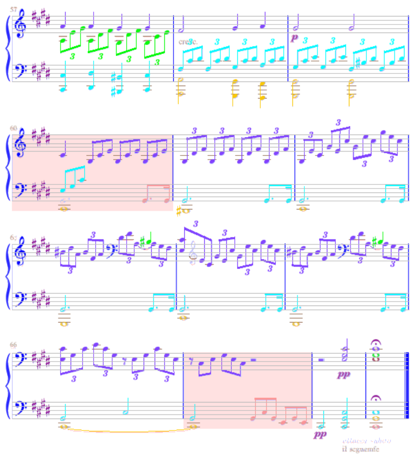


All we have to do now is to delete this false augmentation dot.

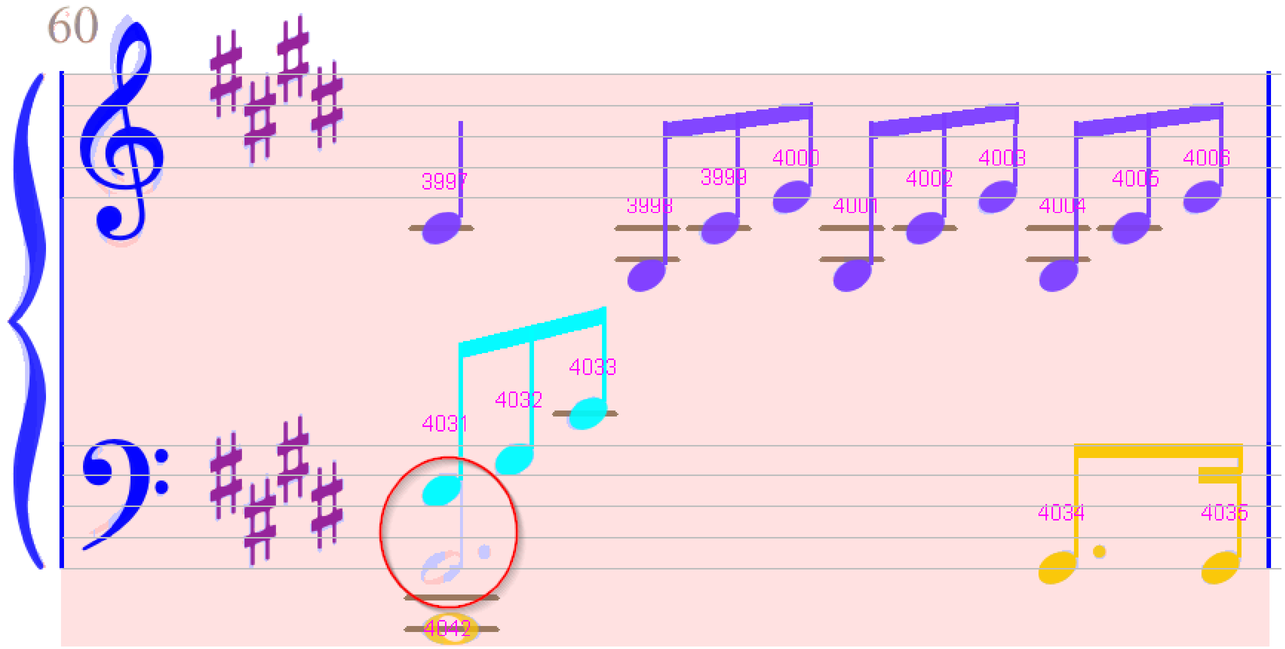


Sheet #5

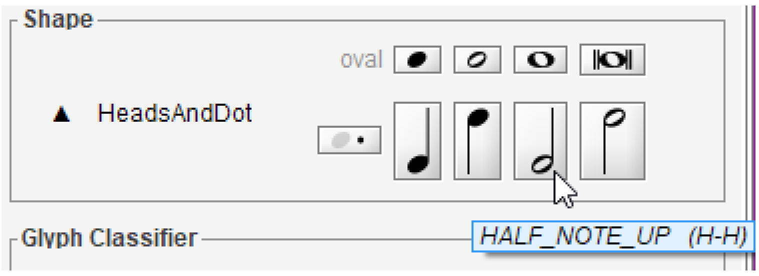
Two measures to fix.

Sheet ID	Raw results
Sheet#5	

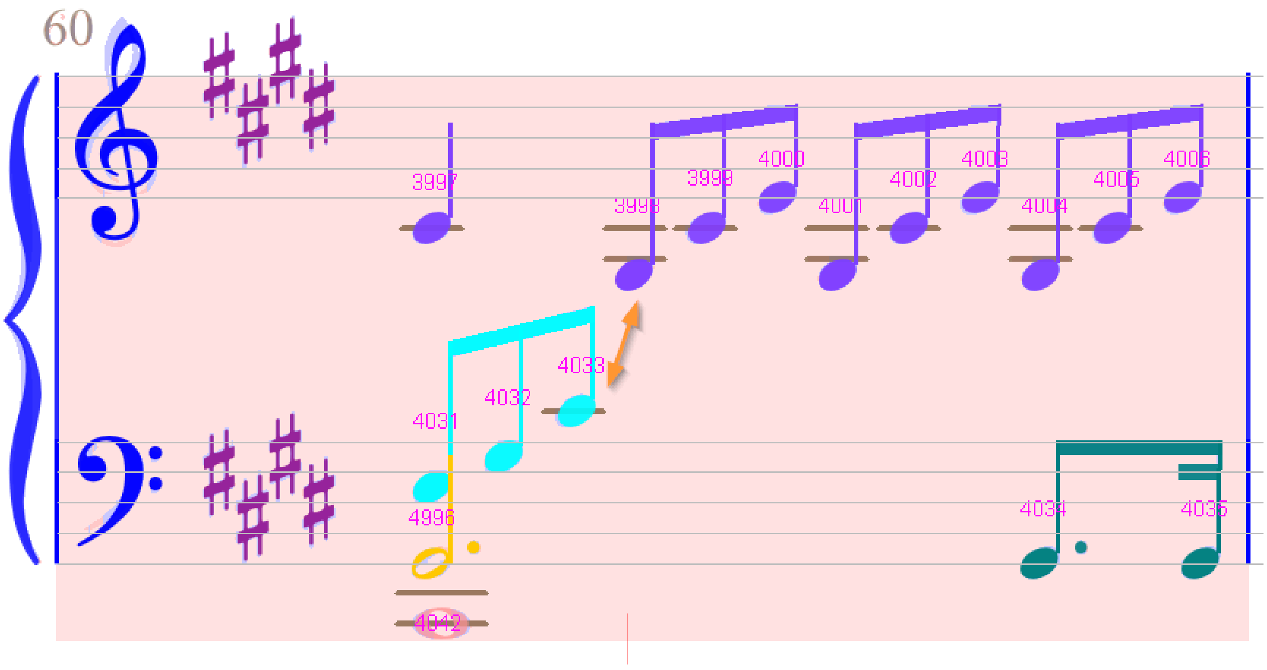
Measure #4



A dotted half-note is missing. We can drag a half-note from the Shape board:

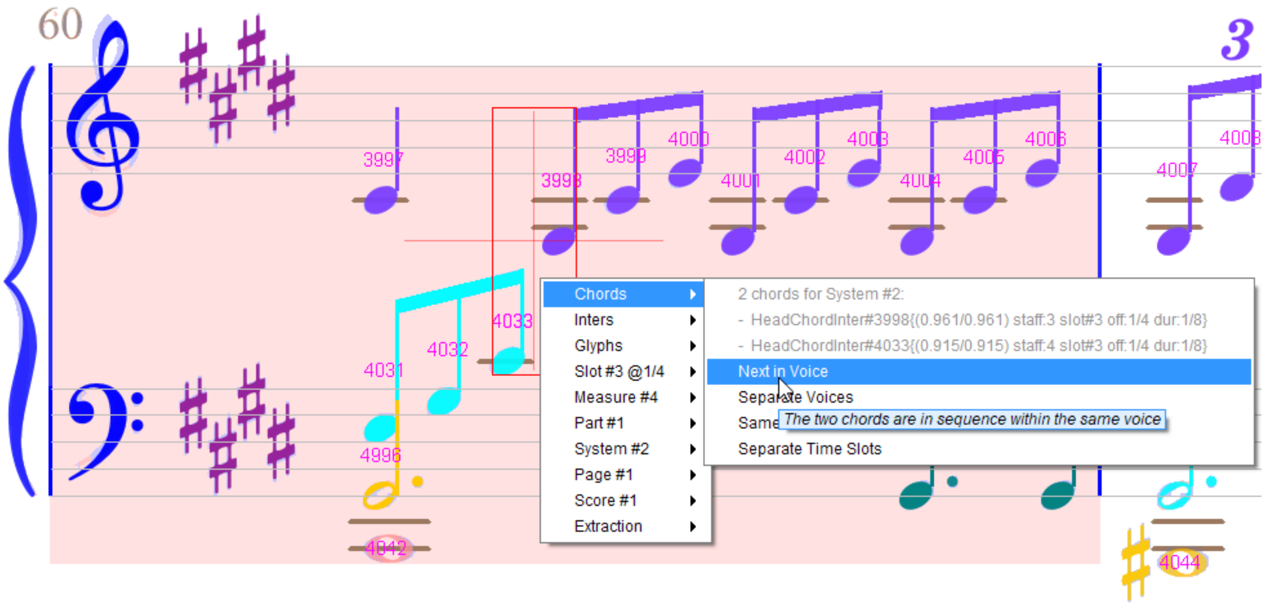


And assign the augmentation dot.



The dotted half-note is now fixed, but we still have no implicit tuplets.

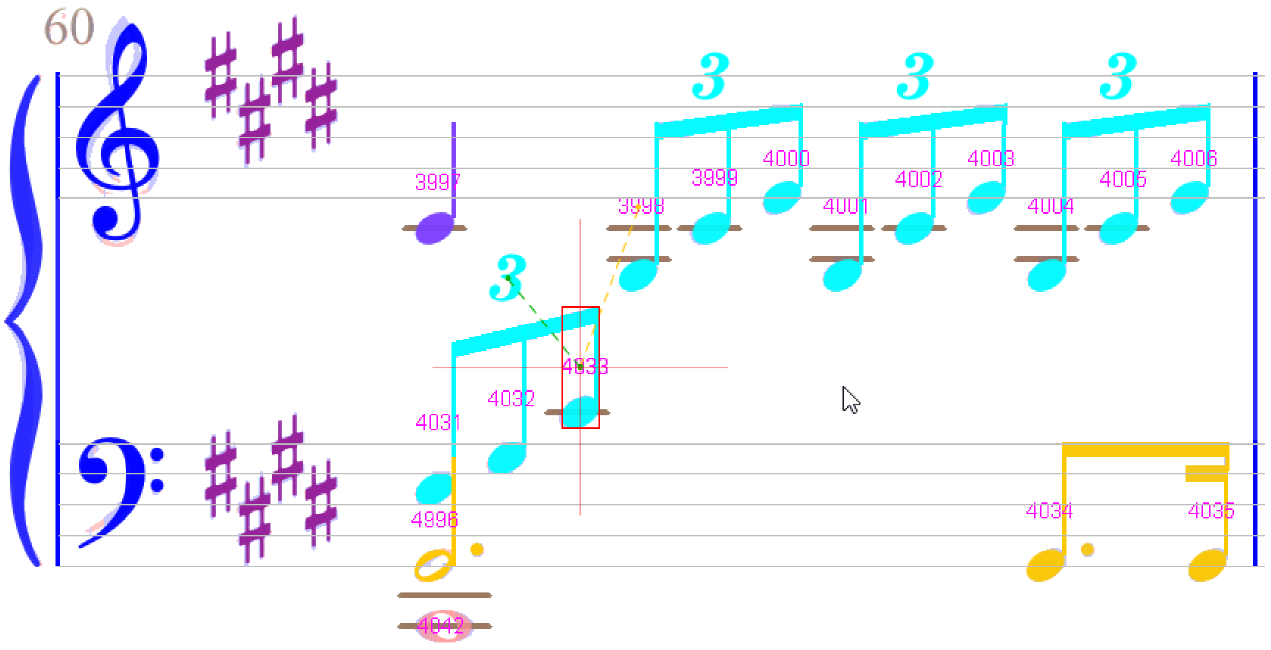
We could manually insert them, but there is an easier way: by voice chaining the beamed notes on the left of the lower staff with the beamed notes on the upper staff.



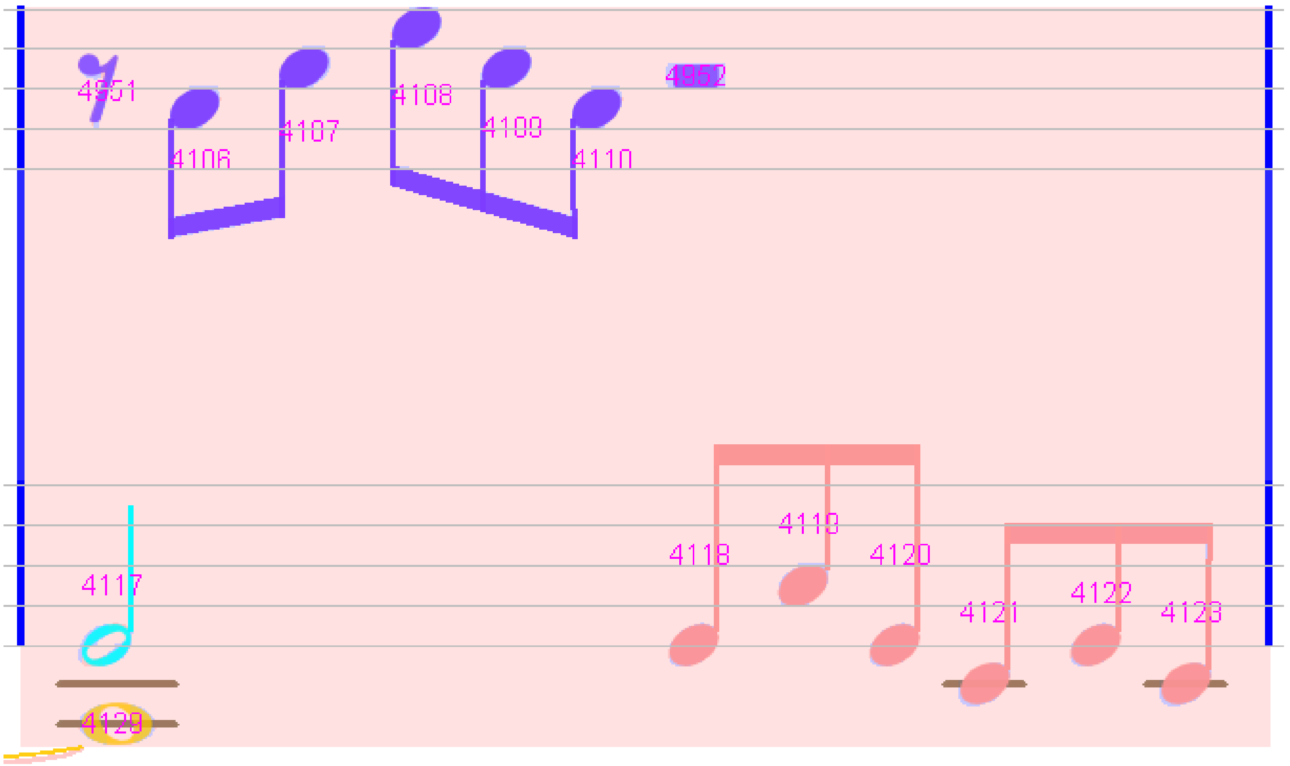
```
2 chords for System #2:
- HeadChordInter#3998{(0.961/0.961) staff:3 slot#3 off:1/4 dur:1/8}
- HeadChordInter#4033{(0.915/0.915) staff:4 slot#3 off:1/4 dur:1/8}
```

The two chords are in sequence within the same voice

And voila:

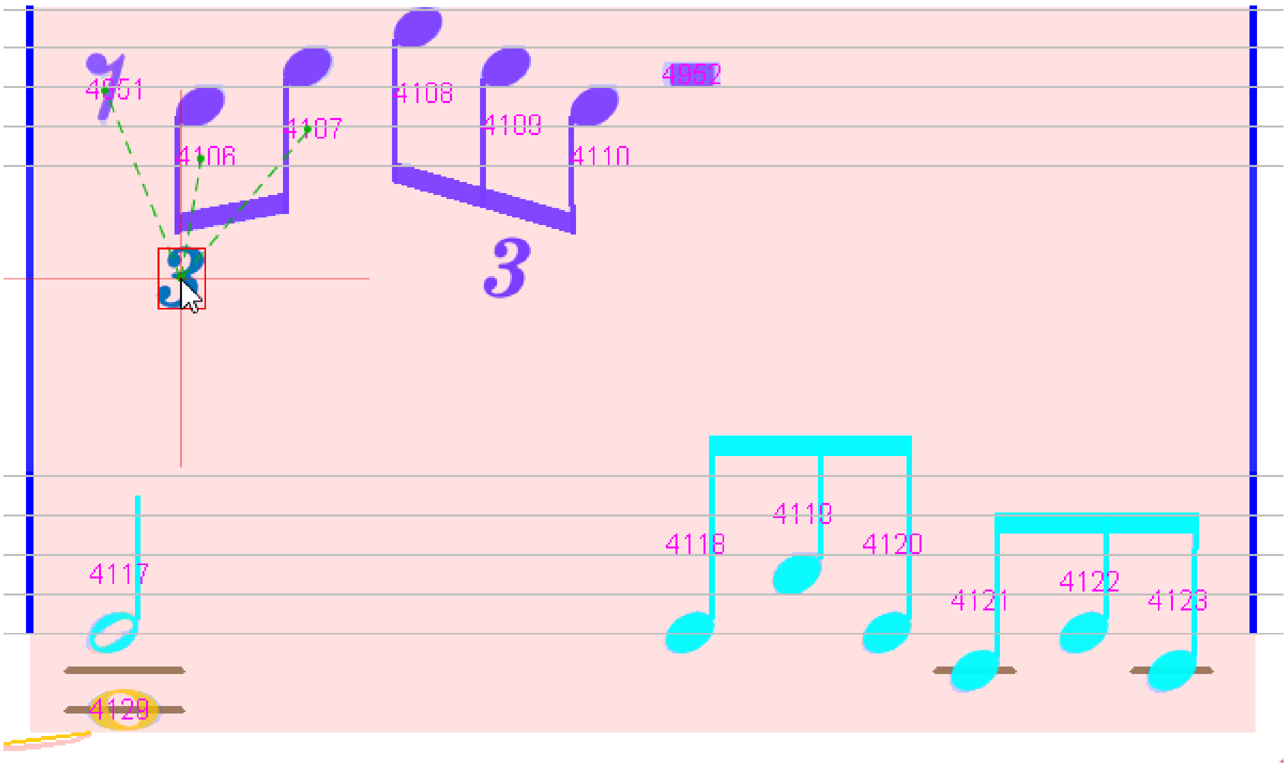


Measure #11



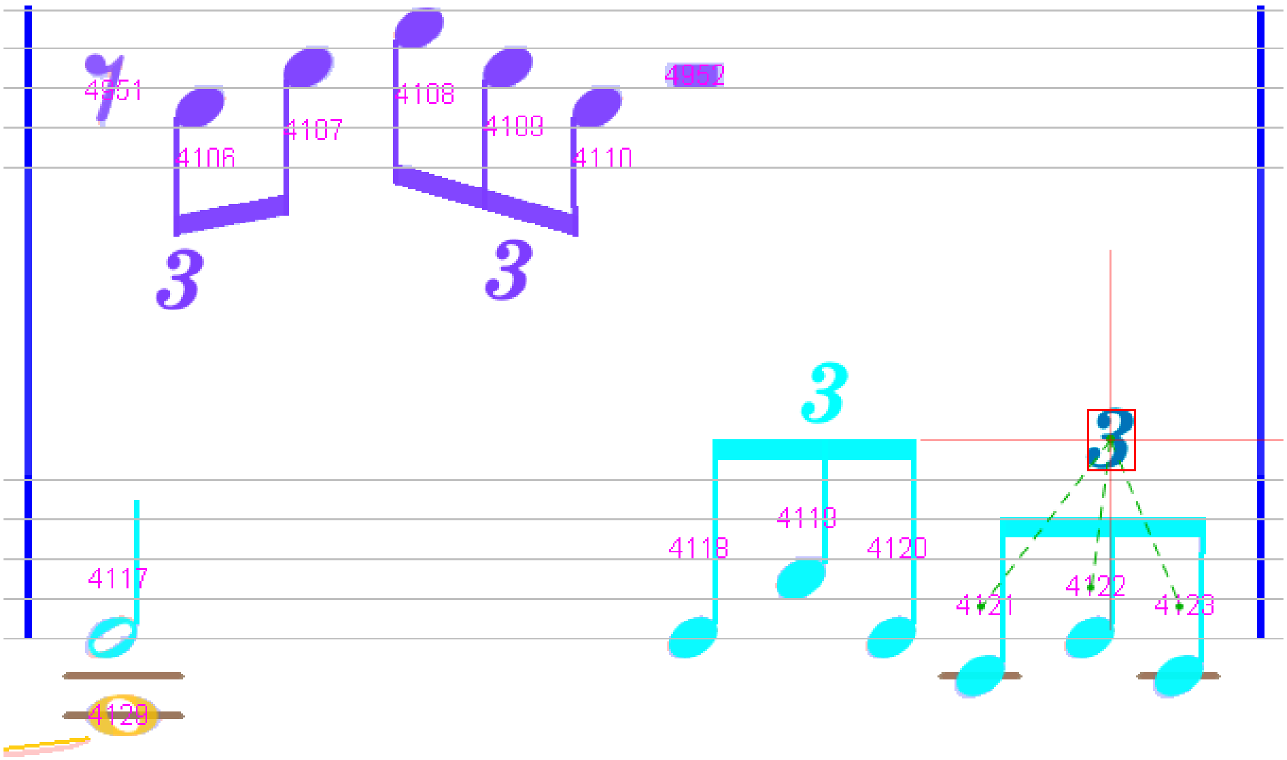
No implicit tuplet has been detected.

We have to insert them manually, from left to right.



The first manual tuplet has been automatically followed by an implicit one.

We finally insert the two missing tuplets on the second staff.



Conclusion

A few remarks, to conclude this editing session.

Regarding precise **tuplet location**:

- We cannot modify the location of an implicit tuplet sign.
  We can put the sign in `Edit` mode and shift it, but as soon as we leave the `Edit` mode, the containing measure is re-processed, resulting in the removal of all implicit tuplets and the creation of new ones at the same locations…
- To actually modify a tuplet location, we have to manually insert a tuplet – typically over the implicit one. This results in a manual (non-implicit) tuplet at the desired location.
  Moreover, this manual tuplet can be further edited if needed.

Regarding **voice number**:

- We have seen how to generally guide the rhythm algorithm – see this section above in this chapter.
- The action `Next in voice` can work only within the same system, but can cross measure boundaries.
- The link it establishes is dynamic, from the left partner to the right partner. In other words, if we change the voice number in a measure, it will impact the linked voice in the following measure.
- Between systems (and *a fortiori* between pages or sheets), the action `Next in voice` cannot apply, only the static action `Preferred voice` is available.

Regarding the **implicit tuplets**:

- The raw transcription of this “Moonlight Sonata” resulted in:
  - 68 measures processed (all requiring implicit tuplets),
  - 3 measures in error because of mis-recognized symbols (we don’t count them),
  - 5 measures in error because of missing tuplets,
  - 60 measures OK with implicit tuplets
    which gives a success rate of 60/65, that is 92%.
- The “implicit tuplets” algorithm can still be improved, but it is usable right now.
  It is recommended to reserve it for selected scores and to be prepared for manual corrections as we did in this session.

Specific features

Aimed at the end user, this chapter gathers features that deserve a specific description.

These are generally new features and are therefore presented here in reverse chronological order.

TABLE OF CONTENTS

Metronome

The picture below displays a typical metronome mark :

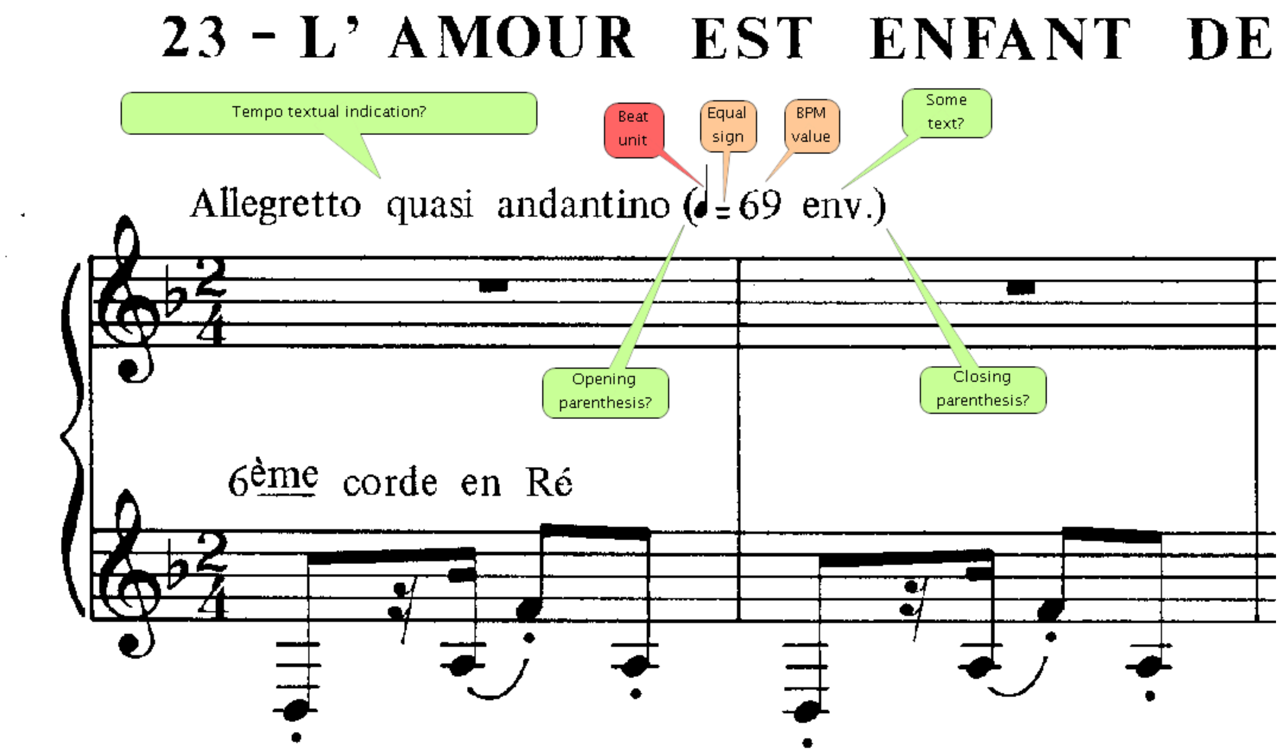


TABLE OF CONTENTS

Structure

From left to right, a metronome mark can contain: [1]

Mandatory?	Item
-	a tempo textual indication, like “Allegretto quasi andantino”
-	an opening parenthesis `(`
YES	a beat unit symbol, like “♩”
YES	the equal (`=`) character
-	some text, like “ca.”
YES	an integer value
-	a minus (`-`) character, followed by a second integer value
-	some text, like “env.”
-	a closing parenthesis `)`

Engine recognition

The OCR can recognize text characters, words and sentences.

The main problem comes from the *beat unit symbol*, like the quarter symbol in the example above, because there is no way to make Tesseract OCR recognize this symbol as a known character. [2]

For the time being, the OMR engine uses a rather acrobatic approach:

1. It looks for an OCR’d sentence which matches a regular expression crafted for the optional and mandatory items listed above,
2. It isolates the word located just before the `=` character[3] and grabs its underlying glyph,
3. It submits this glyph to the glyph classifier, looking for a beat unit symbol.

If the test is positive, then the sentence is recognized as a metronome sentence. As any sentence, the metronome mark is made of words, but one of them is special, it’s the beat unit word which contains just the beat unit symbol.

MusicXML export

The various metronome items (textual tempo, beat unit, bpm values, …) are exported in a MusicXML `direction` element including a `tempo` element.

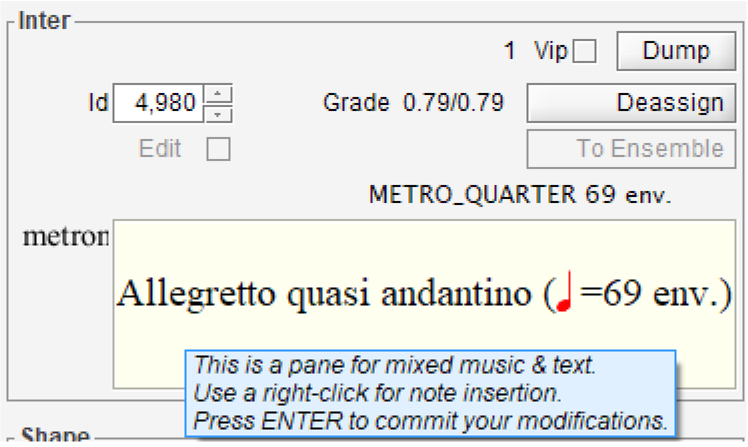
The tempo value is the number of *quarters* per minute, which is determined according to:

- The beat unit,
- The number of beats per minute (either the single value or the mean value if two values are found).

Manual editing

In Audiveris, a standard sentence content cannot be edited directly. The editing can be made at word level only.
But specially for the metronome sentence, editing can be made at both sentence and word levels.

For instance, we can click on any word of the metronome sentence, it gets displayed in the `Inter` board. There, we can click on the `To Ensemble` button to get to the containing sentence: the metronome mark.



Id 4,980

Grade 0.79/0.79

METRO_QUARTER 69 env.

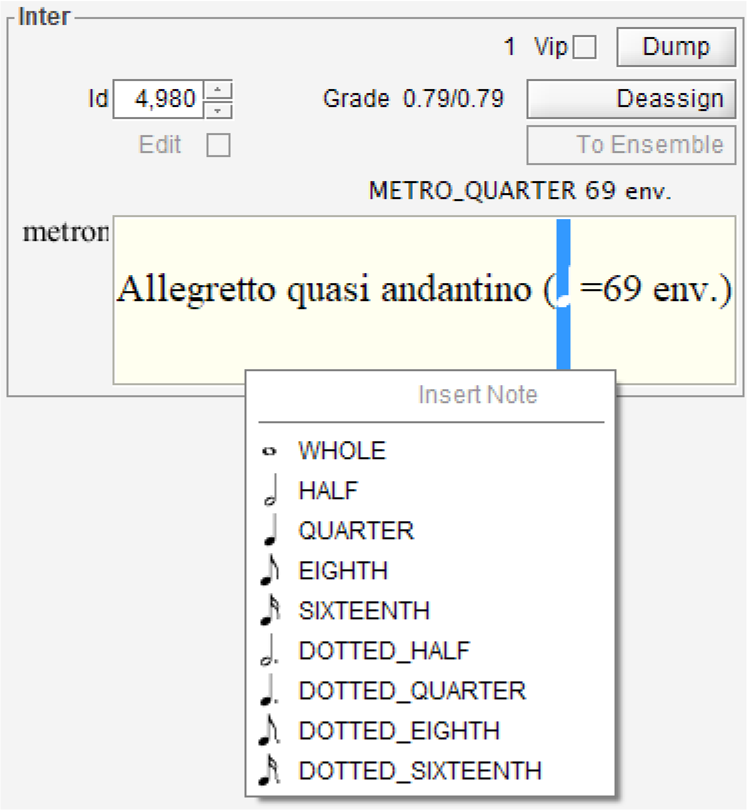
Allegretto quasi andantino (♩=69 env.)

This is a pane for mixed music & text.
Use a right-click for note insertion.
Press ENTER to commit your modifications.

There, we have a small pane where we can see and directly edit the metronome content. The field accepts standard cut and paste actions on text and music characters.

Text can be entered by typing on the keyboard, but music characters are different. They are displayed in red to make them stand out.

To enter a beat unit, we use a right-click in the pane. This opens a popup menu with all the supported beat units:



The selected beat unit is inserted at the current position – or replaces the selected characters if any were selected.

As usual, the modifications get committed when we press the `ENTER` key.
And we can always undo/redo such actions.

Changing a sentence to metronome

The OMR engine may have mis-recognized a metronome mark, and consider it as a **plain sentence**, for example with a *direction* role.

In that case, we can simply select this sentence and manually change its **role** to the `Metronome` role.

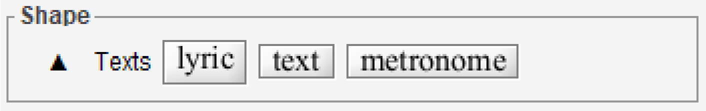
The `Inter` board will consequently display the metronome mark, based on the recognized items. Typically, we'll have to manually correct or insert items like the beat unit, the `=` character, …

The OCR engine may also have totally **ignored the sentence**, for whatever reason.
In that case we can select the sentence (actually we select the underlying glyphs as a whole) and assign it the Metronome **shape**.
And, here again, we manually correct the metronome mark.

Creating from scratch

We can always create a metronome mark from scratch.

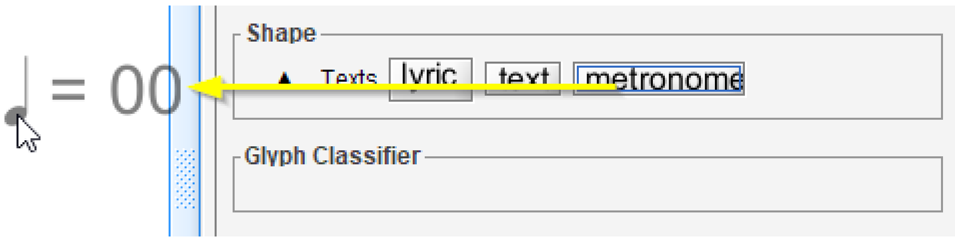
To do so, we click on the `Texts` set in the shape board, which opens this set as:



This set gathers just the text-oriented shapes: `lyric`, (plain) `text` and `metronome`.

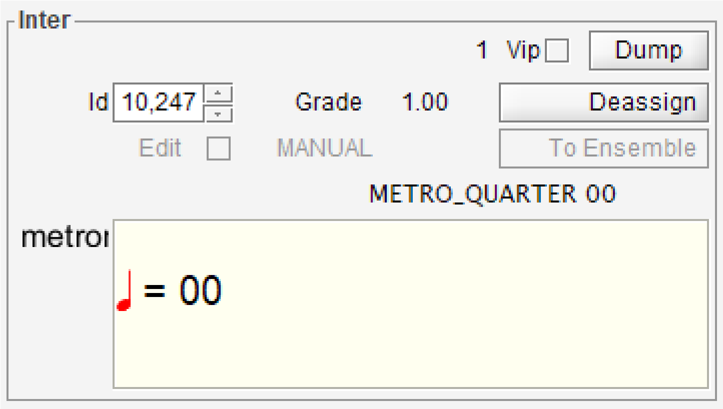
We now can make a drag n’ drop from the metronome button to the sheet location of our choice.

Notice, as we drag away from the `metronome` button and enter the sheet view, that the “metronome” label is replaced by a metronome symbol:



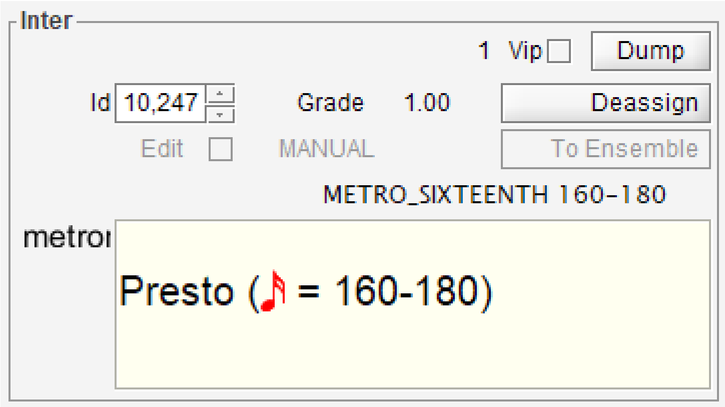
We can drop the symbol, this creates a **minimum** metronome mark, using default elements, at the drop location.

The `Inter` board now displays this minimum metronome mark:



Id 10,247 Grade 1.00
MANUAL
METRO_QUARTER 00
metroı ♩ = 00

We finally have to modify/augment this basic version to get something more in line with what we want. For example:



Inter

1 Vip □ Dump

Id 10,247 Grade 1.00 Deassign

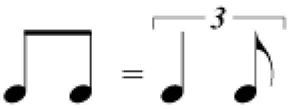
Edit □ MANUAL To Ensemble

METRO_SIXTEENTH 160-180

metror

Presto (♪ = 160-180)

Footnotes

[1] The rhythm-oriented marks, like this one:  are not supported yet. ↩

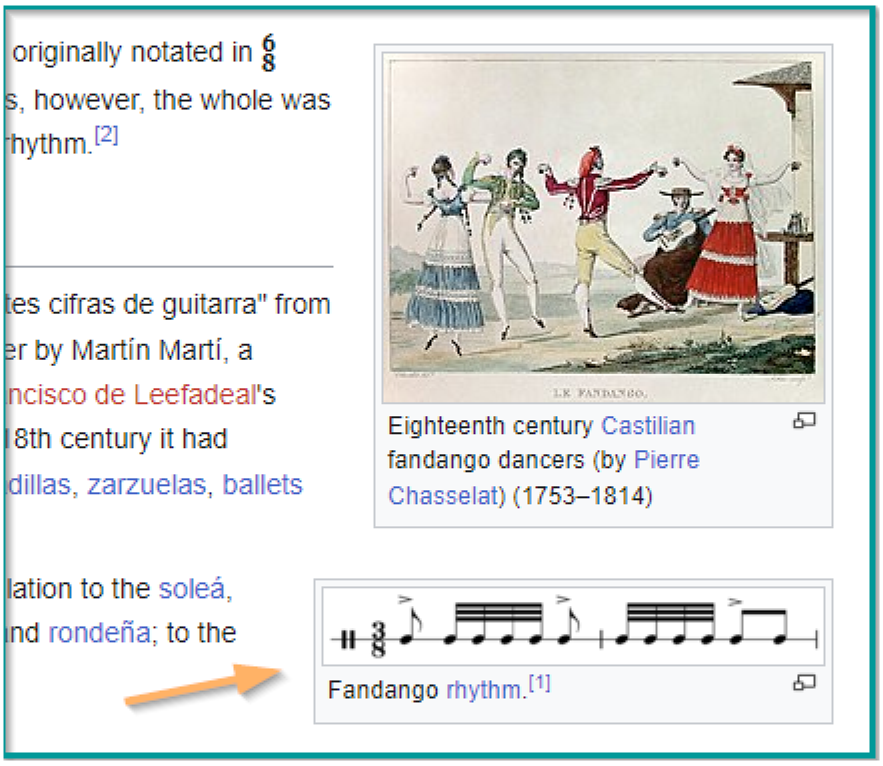
[2] If you know of a way to train Tesseract on beat units, please post a message on the project discussions. ↩

[3] The equal character (`=`) is sometimes OCR’d as a colon character (`:`), so the regular expression has been adjusted to also accept the colon character. ↩

How-to guides / Specific features / Snippets

Snippets

Tiny pieces of music can be found on the Internet, generally for illustration purpose. For example, this Wikipedia page is about the Fandango dance. It provides some explanation text, a picture of fandango dancers and a snippet of music score to describe the specific fandango rhythm:



originally notated in $\frac{6}{8}$
s, however, the whole was
hythm.[2]

tes cifras de guitarra" from
er by Martín Martí, a
ncisco de Leefadeal's
8th century it had
dillas, zarzuelas, ballets

LE FANDANGO.

Eighteenth century Castilian fandango dancers (by Pierre Chasselat) (1753–1814)

lation to the soleá,
nd rondeña; to the

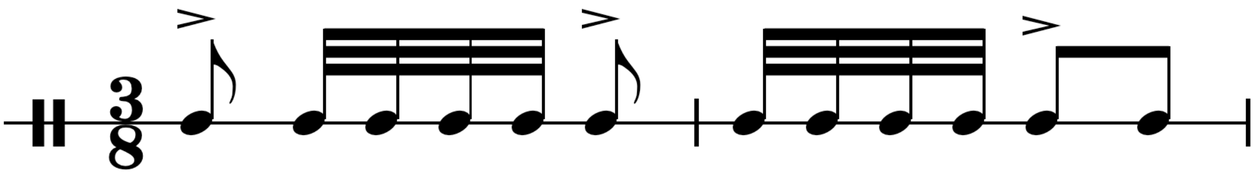
Fandango rhythm.[1]

For the Audiveris OMR engine, the processing of such a tiny score brings a few challenges.

TABLE OF CONTENTS

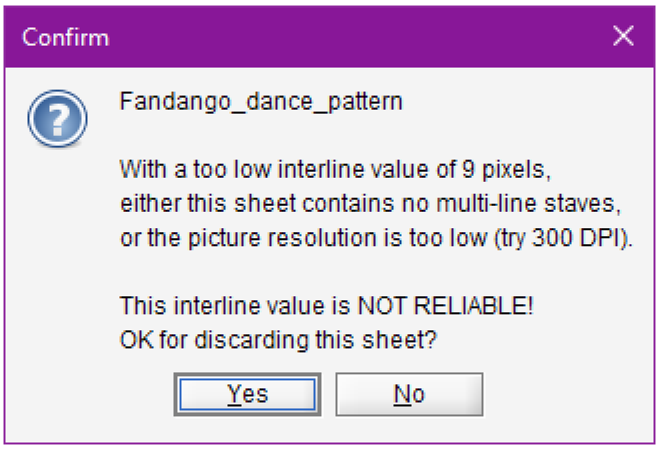
Challenges

Let’s have a closer look at this image (By Hyacinth, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=27409005):



Interline

If we naively launch Audiveris on this input image, we almost immediately get this alert message:



Fandango_dance_pattern

With a too low interline value of 9 pixels,
either this sheet contains no multi-line staves,
or the picture resolution is too low (try 300 DPI).

This interline value is NOT RELIABLE!
OK for discarding this sheet?

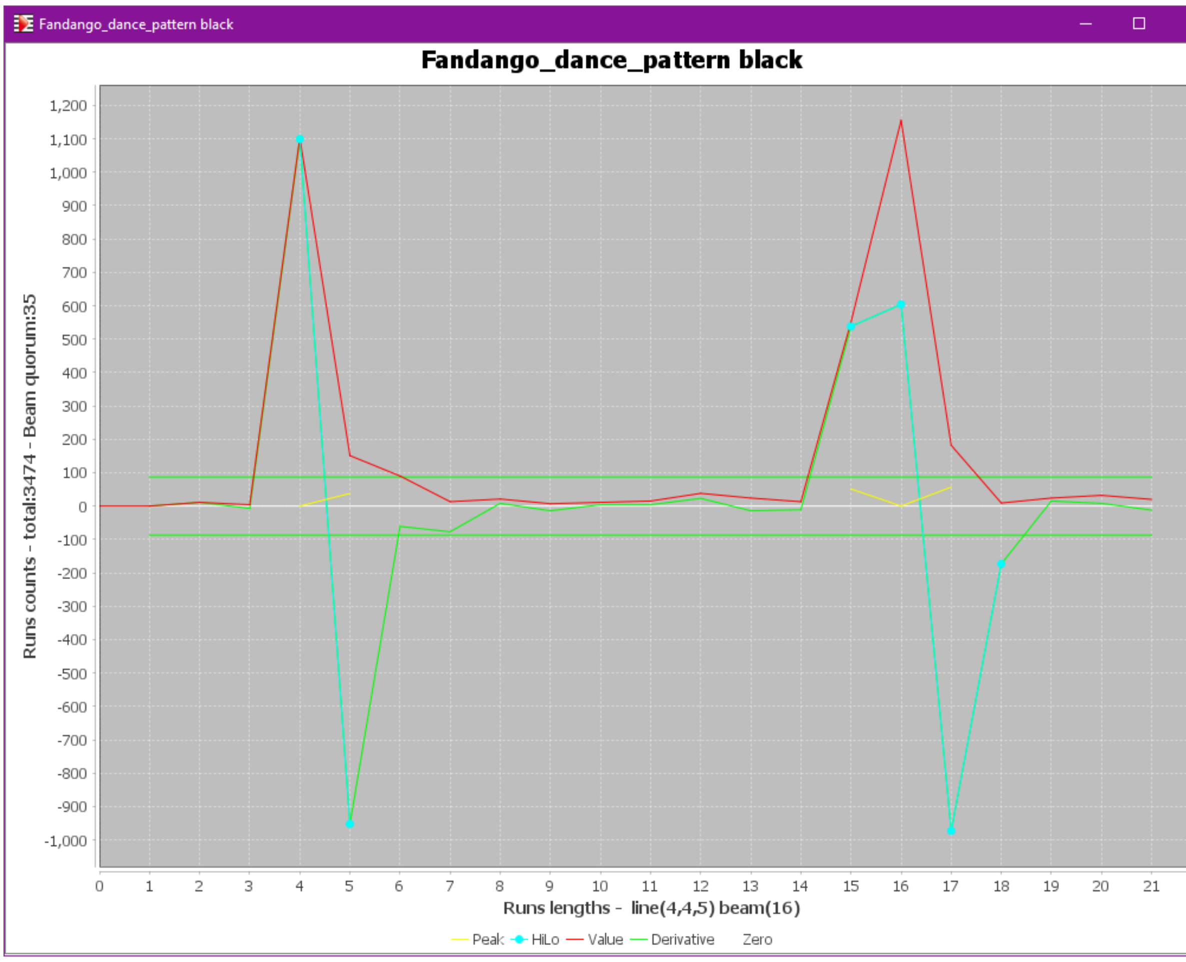
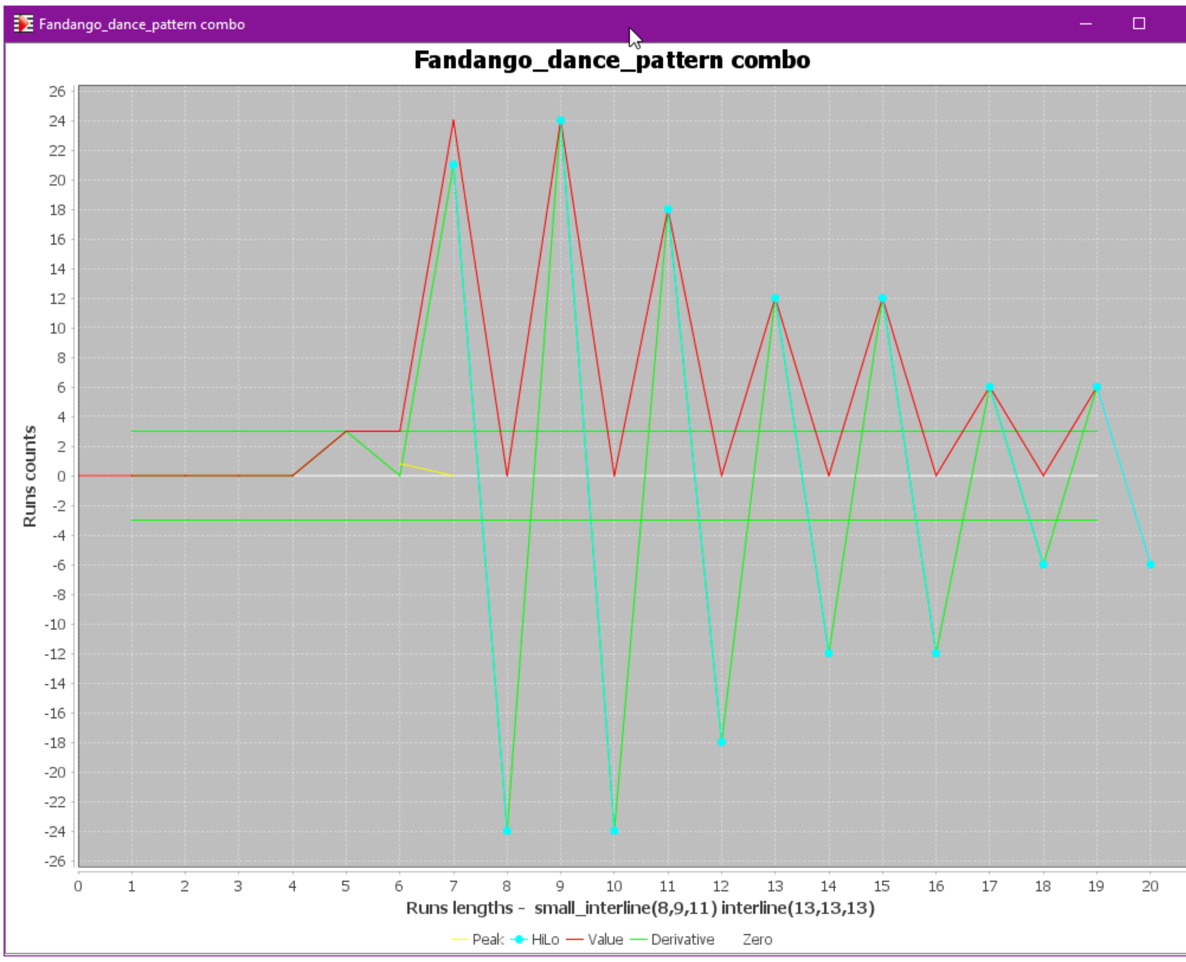
The engine complains about a too low interline value of 9 pixels. This message is issued by the `SCALE` step, whose main task is to retrieve the sheet “interline value”, which is the main vertical distance, in pixels, between two staff lines.

This is a key value, because it determines the general scale of the image, and will drive the following steps such as the heads recognition.

It is retrieved by the `SCALE` step, via the analysis of the so-called “combo histogram”, populated with the vertical runs lengths of the image. The engine measures any vertical black run followed by a vertical white run. The total length is recorded in the combo histogram, which then exhibits a very visible peak at the interline value. For further details, please refer to the Sheet scale chapter.

But in the case of our snippet, the image contains just one staff line. Therefore, there is no physical “interline” the engine could retrieve.

For the sake of completion, here below are the black histogram and the combo histogram:

Legend	Histogram
The black histogram gives correct values for the typical staff line height (4 pixels) and the typical beam height (16 pixels)	
The combo histogram should give the typical interline value, but provides only erratic values	

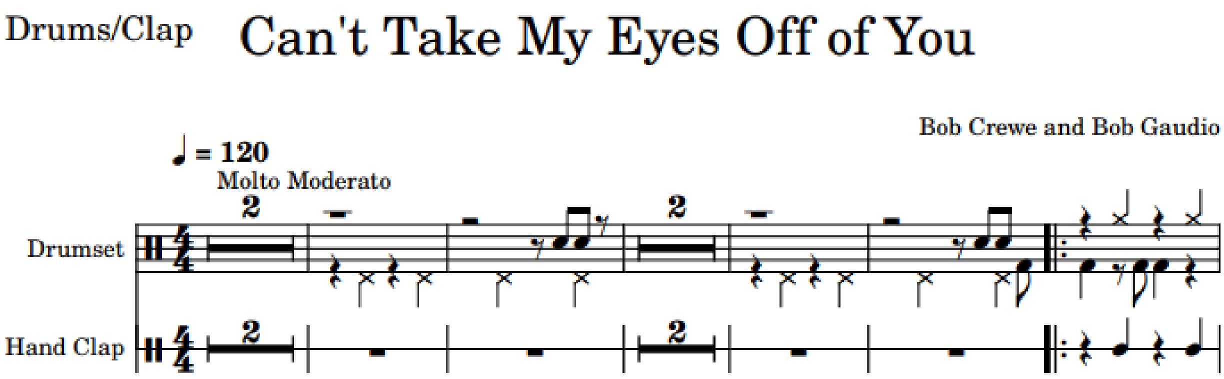
Barline height

Another consequence of the lack of multi-line staves in the input image concerns the typical barline height that the engine should expect.

For a multi-line staff, any barline is expected to go from the top staff line to the bottom staff line. But what if the staff at hand is a 1-line staff? This is no upper line or lower line. So, how far should the OMR engine expect a barline to go above and below the single staff line?

In our snippet example, we have two barlines, one at the center and one at the right side of the staff. They both go from roughly one head height above to one head height below the line. We use the notion of “head height” to replace the lacking notion of interline. These barlines thus have a height of about 2 “interlines”.

But let’s consider this other example, with a 5-line staff and a 1-line staff:

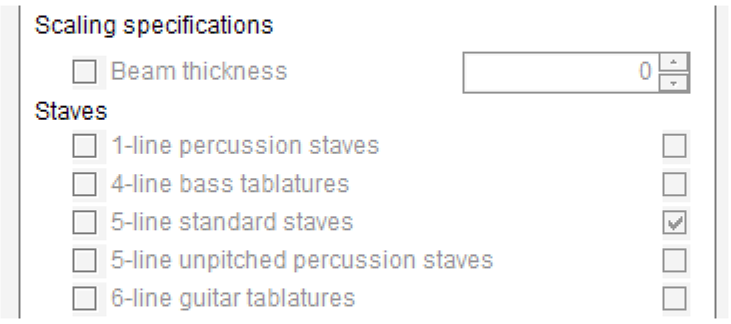
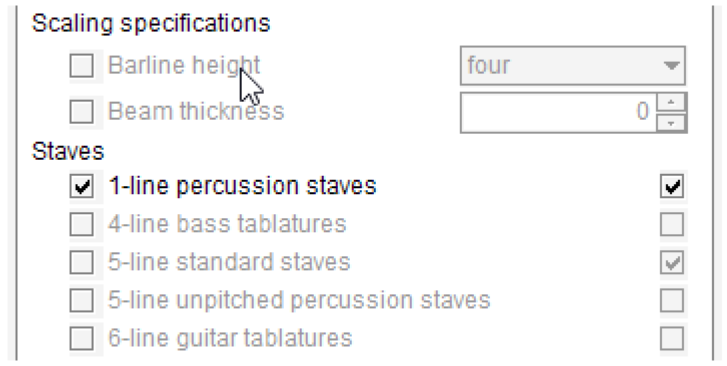
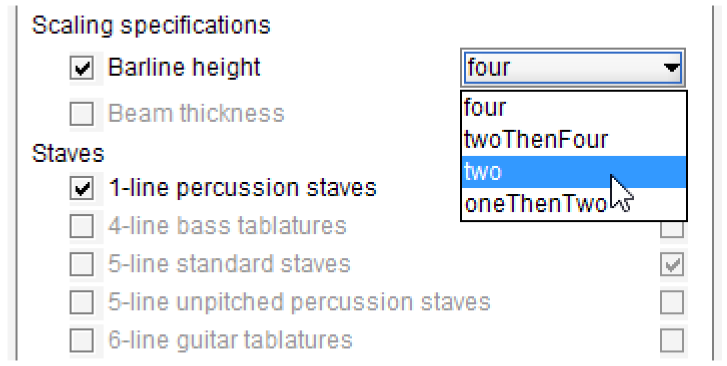
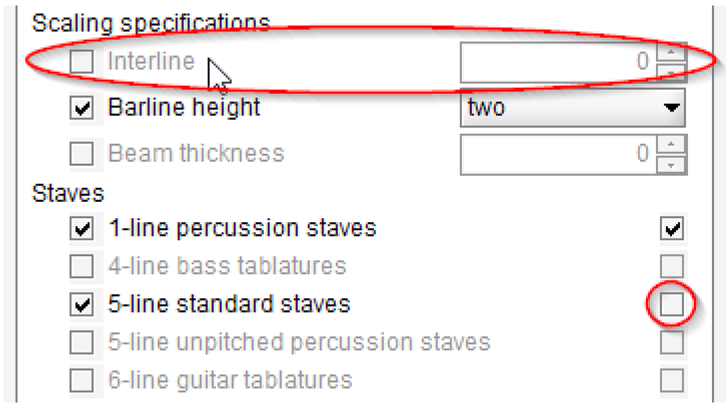
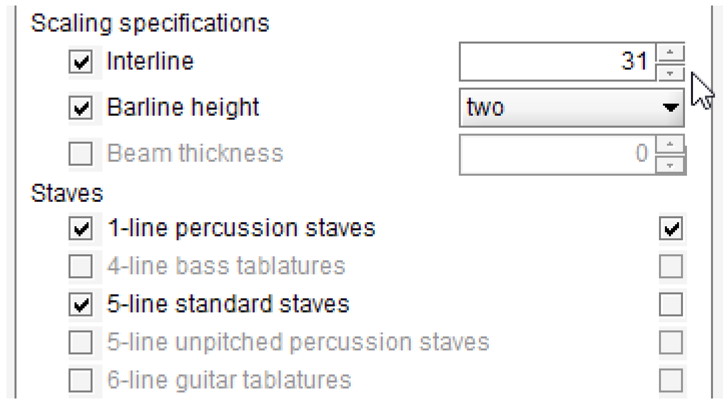


The typical barline height for the 1-line staff is here the same as for the 5-line staff, that is 4 interlines.

Specifications

Since the engine has no way to guess by itself the needed values of interline and barline height, we have to tell the engine which values to use for the image at hand.

This can be performed via the `Book → Set book parameters` menu as follows:

Comment	Dialog
0/ Initial dialog, by default the engine expects standard 5-line staves	 Scaling specifications □ Beam thickness 0 Staves □ 1-line percussion staves □ □ 4-line bass tablatures □ □ 5-line standard staves ☑ □ 5-line unpitched percussion staves □ □ 6-line guitar tablatures □
1/ The 1-line switch has been selected, the “Barline height” field appears	 Scaling specifications □ Barline height four □ Beam thickness 0 Staves ☑ 1-line percussion staves ☑ □ 4-line bass tablatures □ □ 5-line standard staves ☑ □ 5-line unpitched percussion staves □ □ 6-line guitar tablatures □
2/ Barline height set to “two” interlines	 Scaling specifications ☑ Barline height four four twoThenFour two oneThenTwo □ Beam thickness Staves ☑ 1-line percussion staves □ 4-line bass tablatures □ □ 5-line standard staves ☑ □ 5-line unpitched percussion staves □ □ 6-line guitar tablatures □
3/ No multi-line staff in image, the “Interline” field appears	 Scaling specifications □ Interline 0 ☑ Barline height two □ Beam thickness 0 Staves ☑ 1-line percussion staves ☑ □ 4-line bass tablatures □ ☑ 5-line standard staves □ □ 5-line unpitched percussion staves □ □ 6-line guitar tablatures □
4/ Final dialog content	 Scaling specifications ☑ Interline 31 ☑ Barline height two □ Beam thickness 0 Staves ☑ 1-line percussion staves ☑ □ 4-line bass tablatures □ ☑ 5-line standard staves □ □ 5-line unpitched percussion staves □ □ 6-line guitar tablatures □

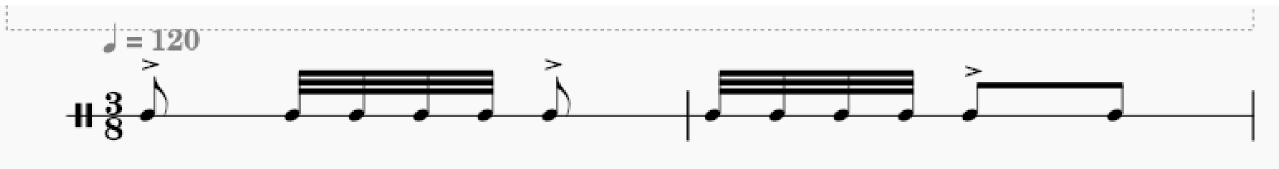
Regarding the “Barline height” field, the dialog prompts for a value among four possibilities:

- `four`: This is the default value of 4 interlines, as seen on typical standard staves. The other possibilities can apply to 1-line staves only.
- `twoThenFour`: Barlines are 4 interlines high, except a half-size for the starting barline (either 2 above or 2 below)
- `two`: All barlines are 2 interlines high, as in our snippet example.
- `oneThenFour`: Barlines are 2 interlines high, except a half-size for the starting barline (either 1 above or 1 below)

Regarding the “Interline” field, it appears when we explicitly tell the engine that there is no standard 5-line staves, and thus there are only 1-line staves in the image. Hence the need for the user to enter an interline value.
We have to measure the typical height of a note head or half the typical height of a barline. We can do so by drawing a lasso either in the Binary tab, or in the Data tab (once the Pixel board has been opened).
This gives about 31 pixels for our snippet example.

Transcription

Once the book parameters are correctly set, the snippet gets perfectly transcribed. Here is the final result using the MuseScore plugin:



Drums UPDATED FOR 5.4

We owe many thanks to Brian Boe for his decisive contribution to this long-awaited functionality.

The first requests for drum notation go back to 2017, see Audiveris issue #33. This delay certainly originated in my personal ignorance (Hervé speaking) about this kind of notation.

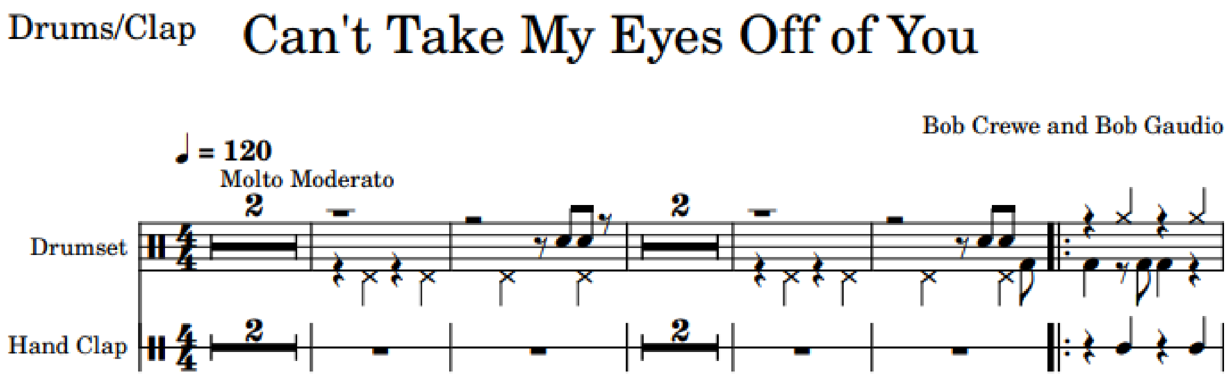
TABLE OF CONTENTS

Example

This Wikipedia article provides a good introduction to drum notation.

Since the 5.3 release, Audiveris supports unpitched precussion notation on both 5-line and 1-line staves, as shown in the excerpt below.

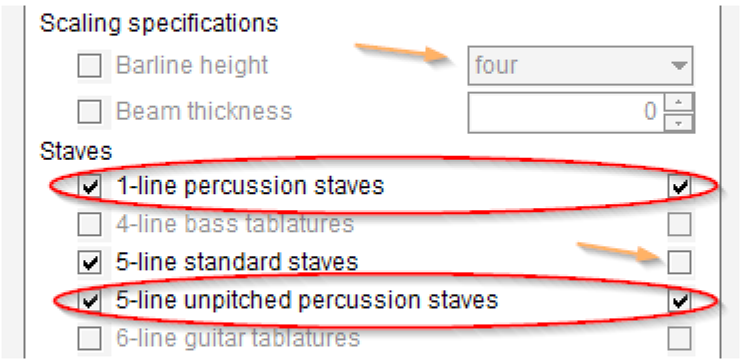
> Note this handbook contains a User editing session dedicated to this precise score example.



1. Each staff starts with a percussion clef,
2. Each individual instrument is defined by a given head motif at a given line position
(drum set mapping).
3. Though not specific to drum notation, we can frequently observe multi-measure rests as well as measure-repeat signs.

Mandatory processing switches

To be able to transcribe the example above, we have to set two specific processing book parameters, available in the dialog `Book → Set book parameters`, in the section named “Staves”, one for 1-line and one for 5-line percussion stave sizes:



Scaling specifications

□ Barline height four

□ Beam thickness 0

Staves

☑ 1-line percussion staves ☑

□ 4-line bass tablatures □

☑ 5-line standard staves □

☑ 5-line unpitched percussion staves ☑

□ 6-line guitar tablatures □

In batch mode, we can also set these switches at the command line interface level:

```
-constant org.audiveris.omr.sheet.ProcessingSwitches.oneLineStaves=true
```

```
-constant org.audiveris.omr.sheet.ProcessingSwitches.drumNotation=true
```

Additional switches

Beside the two mandatory switches, it is worth paying attention to two other lines:

- One is the “**Barline height**”, located just above, in the “Scaling specifications”. In fact, this line is usually hidden. It gets visible only when we set the “1-line percussion staves” switch to on.
  On a multi-line staff, a barline is expected to go from the top line to the bottom line. But on a 1-line staff, what should be the height of barlines? In the example at hand, as in many other examples, the barlines exhibit the same height, whether they belong to a 5-line staff or to a 1-line staff, that is four interlines which is the default value.

- Another one is the processing switch named “**5-line standard staves**”. Since there are no such staves in the example, we can explicitly tell the engine that there are none.
  It has no concrete impact on the example at hand, because there are other multi-line staves available for the OMR engine to measure the score interline value.
  This switch will get more important when we address the case of Snippets.

Drum set mapping

There seems to be no universal specification for drum set mapping, only some recommendations.
This means that, depending on score author, the mapping can be different.

Audiveris provides a default mapping via the `drum-set.xml` file in its `res` resource folder.
The content of this XML file is listed in the appendix.

Within the `drum-set` XML root element, there is one `staff` XML element per staff size (one for 1-line staves, one for 5-line staves).

The mapping definition is a list of entries within the containing `staff` XML element.

- It is presented by increasing line/space number, but only for ease of browsing.
- Each `entry` XML element contains the following XML attributes:
  - ”`pitch-position`”: line/space number (0 being the mid-line, values increasing top down)
  - ”`motif`”: general head shape (oval, small, cross, diamond, triangle, circle)
  - ”`sound`”: intrument name (Acoustic_Bass_Drum, etc. see the appendix)
  - ”`sign`”: this attribute is optional, it indicates a playing technique

(PLAYING_OPEN, PLAYING_HALF_OPEN, PLAYING_CLOSED)

This protected resource is just the *default* mapping:

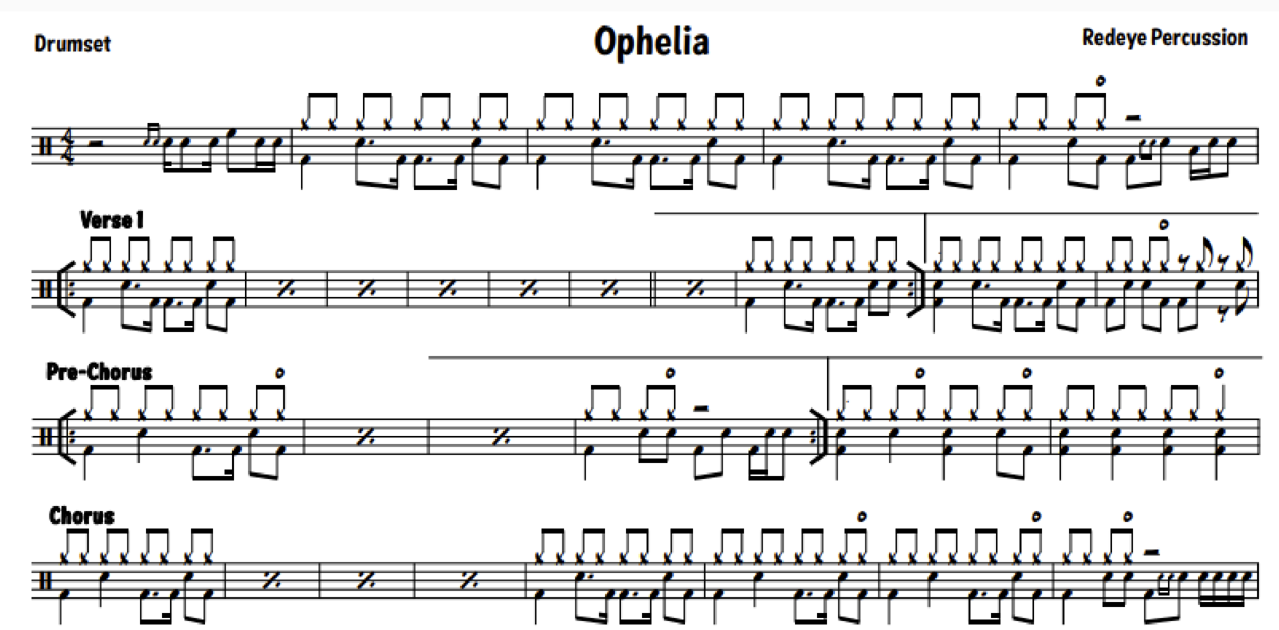
- As an end-user, we can always incrementally override some or all of the default mapping entries, by writing our own entries in a similar but certainly smaller `drum-set.xml` file to be located in the user `config` folder.
- To “nullify” an existing default entry, we simply specify a “`null`” value for its “`sound`” attribute.

Transcription

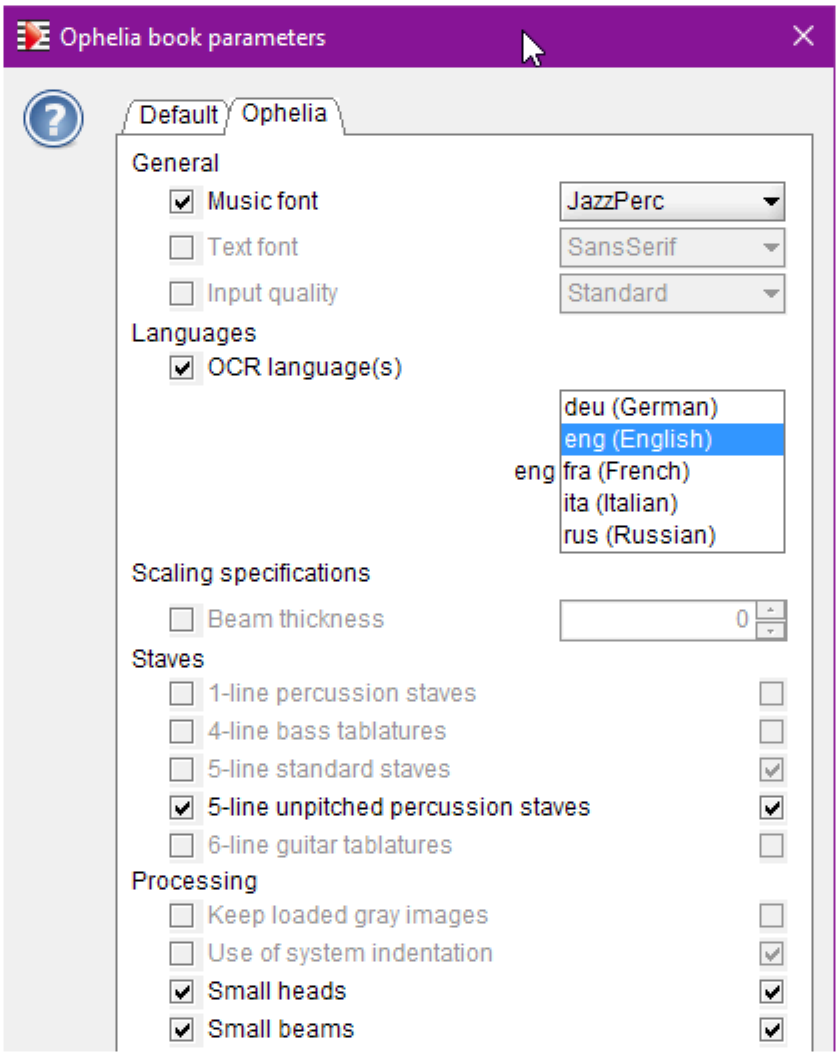
Among the examples used when working on drums with Brian, we can recommend the Redeye Percussion site.
Its “Sheet Music” top link gives access to hundreds of percussion scores.

Here below is the example of Ophelia, a simple one-page score.

The beginning of the PDF file, downloaded from Redeye Percussion site, is as follows:



Before launching the OMR engine, we specify book parameters using the `Book → Set book parameters` dialog:



Ophelia book parameters

Default | Ophelia

General

- ☑ Music font — JazzPerc
- □ Text font — SansSerif
- □ Input quality — Standard

Languages

- ☑ OCR language(s) — eng
  - deu (German)
  - eng (English)
  - fra (French)
  - ita (Italian)
  - rus (Russian)

Scaling specifications

- □ Beam thickness — 0

Staves

- □ 1-line percussion staves □
- □ 4-line bass tablatures □
- □ 5-line standard staves ☑
- ☑ 5-line unpitched percussion staves ☑
- □ 6-line guitar tablatures □

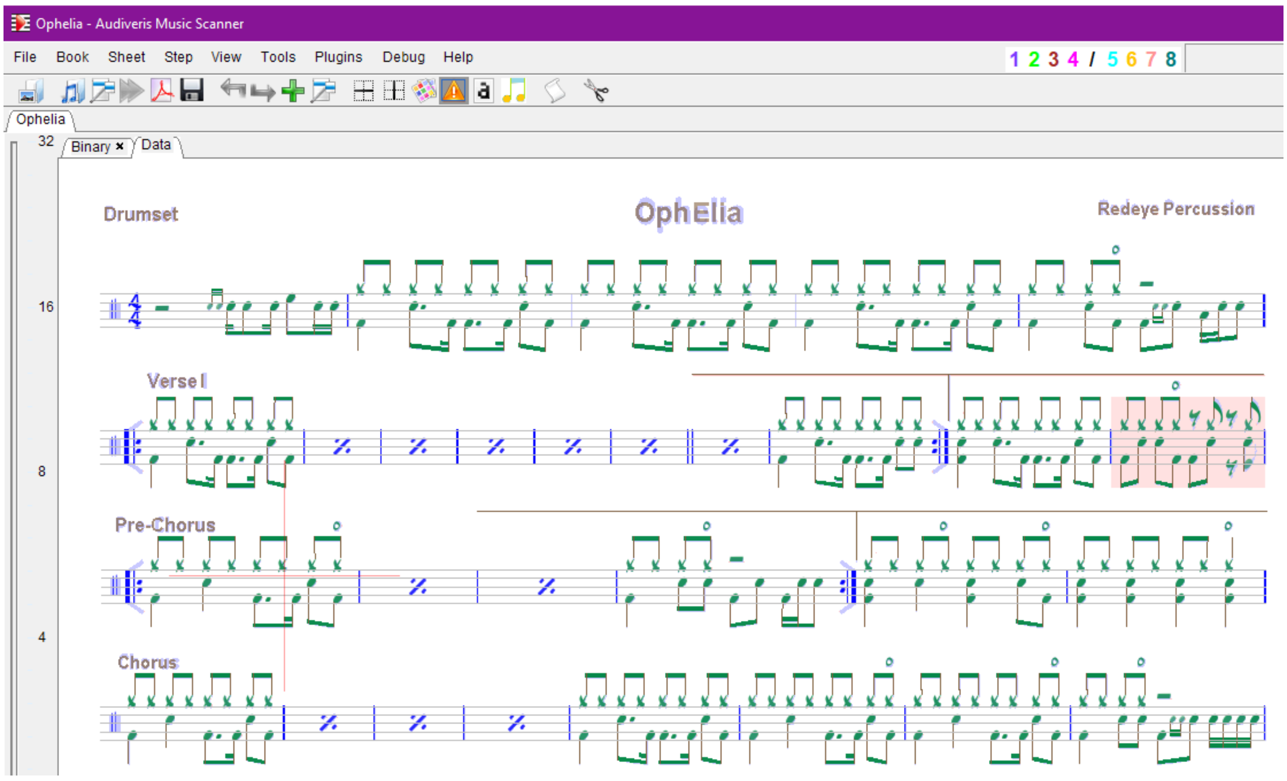
Processing

- □ Keep loaded gray images □
- □ Use of system indentation ☑
- ☑ Small heads ☑
- ☑ Small beams ☑

Note the specific selections made:

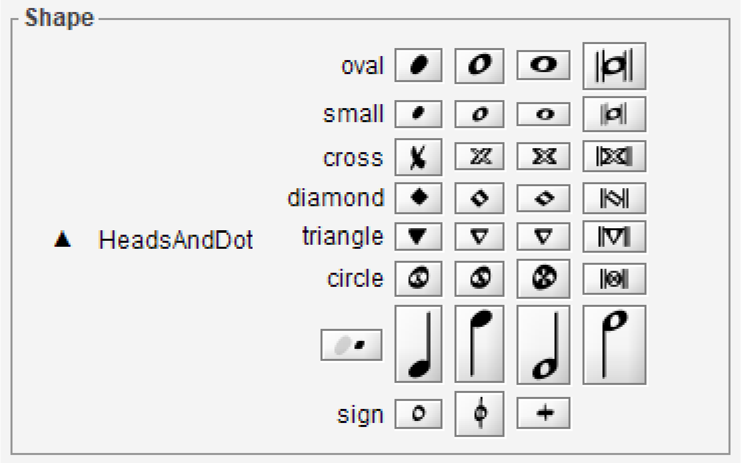
- **Music font**: We do need the “JazzPerc” music font family for best efficiency on this example.
- **5-line unpitched percussion staves**: this will allow the processing of drums notation on 5-line staves
- **Small heads**: we have cue heads in this score
- **Small beams**: we have cue beams tied to the cue heads

And with no additional user action, the raw transcription gives:



There is one abnormal measure, shown in pink.
It is due to a flag mistaken with a black head, something easy to fix.

Should we need to manually fix other transcription errors, we would certainly end up playing with the `HeadsAndDot` palette in the shape board.
For the “`Jazz Perc`” font family and the selected processing switches, this palette provides the following symbols:



It is organized as follows:

- First, the heads part:
  - One row per head **motif** (`oval` through `circle`)
  - One column per head increasing **duration** (1/4, 1/2 , 1, 2)
- The next row is dedicated to the augmentation dot and the pre-built quarter and half notes.
- The last row presents the playing **signs** [1] (open, half-open, closed) that can appear away from heads.

Appendix

Here below is the content of the `drum-set.xml` file provided in the Audiveris `res` application folder.

Remember the end user can still override some default definitions via a similar `drum-set.xml` file to be located in the user `config` folder.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- =========================================================================================== --
<!--                                                                                             --
<!--                                     d r u m - s e t . x m l                                 --
<!--                                                                                             --
<!-- =========================================================================================== --

<!--
    This file, located in the Audiveris 'res' folder, defines the default drum mapping.
    The end-user can override part or all of this mapping by writing a specific drum-set.xml file
    located in the Audiveris user 'config' folder.

    Such a file defines Audiveris entries for general midi percussion keys.
    Definitions are grouped by staff line count with one attribute:

    - line-count        (Mandatory)
                        count of lines in percussion staff
                        (1 or 5)

    Within a staff group, it is written as a list of entries for a sound map organized as:
    key:    a tuple (pitch-position, motif, [sign])
    value:  the corresponding sound value

    - pitch-position:   (Mandatory)
                        head position relative to staff line/space
                        (0 for middle line, positive downwards)

    - motif:            (Mandatory)
                        motif of head shape, regardless of its duration
                        (oval, small, cross, diamond, triangle, circle)

    - sign:             (Optional, default value is null)
                        percussion playing technique
                        (PLAYING_OPEN, PLAYING_HALF_OPEN, PLAYING_CLOSED)
    - sound:            (Mandatory)
                        name of drum sound
                        The name must contain no space, no dash, just underscores.
                        A null sound value removes the entry at the (pitch-position,motif,sign) tup
                        For a comprehensive list of sound names, please refer to
                        https://computermusicresource.com/GM.Percussion.KeyMap.html

    -->

<drum-set>

  <staff line-count="1">

    <!-- -1 -->
    <entry pitch-position="-1" motif="oval"  sound="Hi_Bongo"/>
    <entry pitch-position="-1" motif="cross" sound="Maracas"/> <!-- Surely wrong... -->

    <!-- 0 -->
    <entry pitch-position="0"  motif="oval"  sound="Hand_Clap"/>

    <!-- 1 -->
    <entry pitch-position="1"  motif="oval"  sound="Low_Bongo"/>
    <entry pitch-position="1"  motif="cross" sound="Cowbell"/>

  </staff>

  <staff line-count="5">

    <!-- -8 -->
    <entry pitch-position="-8" motif="cross"   sound="Splash_Cymbal"/>
    <entry pitch-position="-8" motif="circle"  sound="Chinese_Cymbal"/>

    <!-- -7 -->
    <entry pitch-position="-7" motif="cross"   sound="Crash_Cymbal_2"/>
    <entry pitch-position="-7" motif="circle"  sound="Crash_Cymbal_2"/>

    <!-- -6 -->
    <entry pitch-position="-6" motif="cross"   sound="Crash_Cymbal_1"/>
    <entry pitch-position="-6" motif="circle"  sound="Crash_Cymbal_1"/>
    <entry pitch-position="-6" motif="diamond" sound="Crash_Cymbal_1"/>
    <!-- -5 -->
    <entry pitch-position="-5" motif="cross"    sign="PLAYING_OPEN"   sound="Open_Hi_Hat"/>
    <entry pitch-position="-5" motif="circle"                         sound="Open_Hi_Hat"/>
    <entry pitch-position="-5" motif="cross"    sign="PLAYING_CLOSED" sound="Closed_Hi_Hat"/>
    <entry pitch-position="-5" motif="cross"                          sound="Closed_Hi_Hat"/>

    <!-- -4 -->
    <entry pitch-position="-4" motif="oval"     sound="High_Tom"/>
    <entry pitch-position="-4" motif="cross"    sound="Ride_Cymbal_1"/>
    <entry pitch-position="-4" motif="circle"   sound="Ride_Cymbal_1"/>
    <entry pitch-position="-4" motif="diamond"  sound="Ride_Bell"/> <!-- Or Ride_Cymbal_1 or 2 -->

    <!-- -3 -->
    <entry pitch-position="-3" motif="oval"     sound="Hi_Mid_Tom"/>
    <entry pitch-position="-3" motif="cross"    sound="Open_Hi_Hat"/>
    <entry pitch-position="-3" motif="circle"   sound="Open_Hi_Hat"/>
    <entry pitch-position="-3" motif="triangle" sound="Cowbell"/>

    <!-- -2 -->
    <entry pitch-position="-2" motif="oval"     sound="Low_Mid_Tom"/>
    <entry pitch-position="-2" motif="triangle" sound="Cowbell"/>
    <entry pitch-position="-2" motif="diamond"  sound="Tambourine"/>
    <entry pitch-position="-2" motif="cross"    sound="Ride_Cymbal_2"/>
    <entry pitch-position="-2" motif="circle"   sound="Ride_Cymbal_2"/>

    <!-- -1 -->
    <entry pitch-position="-1" motif="oval"     sound="Acoustic_Snare"/>
    <entry pitch-position="-1" motif="small"    sound="Acoustic_Snare"/>
    <entry pitch-position="-1" motif="cross"    sound="Side_Stick"/>
    <entry pitch-position="-1" motif="circle"   sound="Side_Stick"/>

    <!-- 0 -->
    <entry pitch-position="0" motif="oval"      sound="Low_Tom"/>

    <!-- 1 -->
    <entry pitch-position="1" motif="oval"      sound="High_Floor_Tom"/>

    <!-- 2 -->
    <entry pitch-position="2" motif="oval"      sound="Low_Floor_Tom"/>
    <entry pitch-position="2" motif="cross"     sound="Low_Conga"/>
    <entry pitch-position="2" motif="circle"    sound="Low_Conga"/>
    <!-- 3 -->
    <entry pitch-position="3" motif="oval"      sound="Bass_Drum_1"/>

    <!-- 4 -->
    <entry pitch-position="4" motif="oval"      sound="Acoustic_Bass_Drum"/>
    <entry pitch-position="4" motif="cross"     sound="Open_Hi_Conga"/>
    <entry pitch-position="4" motif="circle"    sound="Open_Hi_Conga"/>

    <!-- 5 -->
    <!-- w/ open sign -> foot splash -->
    <entry pitch-position="5" motif="cross"     sound="Pedal_Hi_Hat"/>
    <entry pitch-position="5" motif="circle"    sound="Open_Hi_Hat"/>

<!--
    Here below is the list of not yet assigned sounds
    To actually assign one sound, uncomment the line and replace the "null" values
    by actual pitch-position and motif.

    <entry pitch-position="null" motif="null" sound="Hand_Clap"/>
    <entry pitch-position="null" motif="null" sound="Vibraslap"/>
    <entry pitch-position="null" motif="null" sound="Electric_Snare"/>
    <entry pitch-position="null" motif="null" sound="Hi_Bongo"/>
    <entry pitch-position="null" motif="null" sound="Low_Bongo"/>
    <entry pitch-position="null" motif="null" sound="Mute_Hi_Conga"/>
    <entry pitch-position="null" motif="null" sound="High_Timbale"/>
    <entry pitch-position="null" motif="null" sound="Low_Timbale"/>
    <entry pitch-position="null" motif="null" sound="High_Agogo"/>
    <entry pitch-position="null" motif="null" sound="Low_Agogo"/>
    <entry pitch-position="null" motif="null" sound="Cabasa"/>
    <entry pitch-position="null" motif="null" sound="Maracas"/>
    <entry pitch-position="null" motif="null" sound="Short_Whistle"/>
    <entry pitch-position="null" motif="null" sound="Long_Whistle"/>
    <entry pitch-position="null" motif="null" sound="Short_Guiro"/>
    <entry pitch-position="null" motif="null" sound="Long_Guiro"/>
    <entry pitch-position="null" motif="null" sound="Claves"/>
    <entry pitch-position="null" motif="null" sound="Hi_Wood_Block"/>
    <entry pitch-position="null" motif="null" sound="Low_Wood_Block"/>
    <entry pitch-position="null" motif="null" sound="Mute_Cuica"/>
    <entry pitch-position="null" motif="null" sound="Open_Cuica"/>
    <entry pitch-position="null" motif="null" sound="Mute_Triangle"/>
    <entry pitch-position="null" motif="null" sound="Open_Triangle"/>
    -->
```

```
  </staff>

</drum-set>
```

Footnotes

[1] These playing signs are not heads *per se*, they are plain symbols recognized via the glyph classifier during the SYMBOLS step. ↩

Fonts SINCE 5.3

Audiveris uses specific fonts for music symbols and for text symbols.

TABLE OF CONTENTS

Music fonts

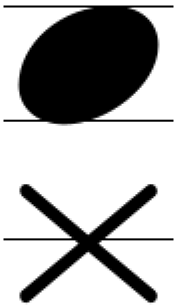
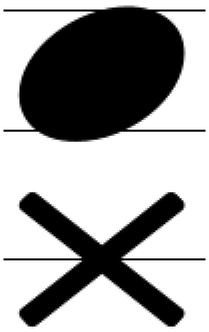
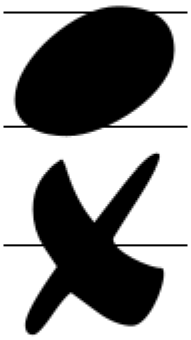
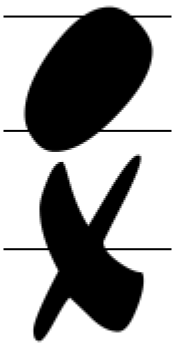
Before 5.3 release, only one music font family was used (**Musical Symbols** font initially, later replaced by **Bravura** font for SMuFL compliance).

Since 5.3 release, we can tell Audiveris which music font family to use by default, or for a given book or sheet.

Head symbols

This choice is key for **head symbols**, since the OMR engine uses a *template matching* technique to detect them during the HEADS step.

Here below are samples of just two significant head shapes – black head and cross head – for each of the available music font families (**Bravura** (the default), **Leland**, **Finale Jazz** and **Jazz Perc**).
Pictures are magnified 4 times for a better reading:

Shape / Font Family	Bravura	Leland	Finale Jazz	Jazz Perc
Black Head and **Cross Head**				

The head templates are built upon a specific font:

- its family is the user-selected music font family,
- its precise size is derived from the measured staff interline.

> **IMPORTANT**
> Due to the way template matching works, it is essential to carefully check the match between the input score image and the selected font family.

- The major difference is between *standard* fonts (`Bravura` and `Leland`) and *jazz* fonts (`Finale Jazz` and `Jazz Perc`), especially for all the cross-like shapes.
- Within the jazz fonts, `Finale Jazz` and `Jazz Perc` differ mainly by their width, and the impact on template matching is more visible on oval-like shapes.

Other symbols

For symbols other than head symbols, such as Clef, Flag, etc, recognition is not based on template matching but on a neural network.

With proper training on representative samples, the neural network is able to recognize their shape, regardless of the selected music font family.
We can observe its top 5 results in the `Glyph Classifier` board.

At this point, the reader may wonder why this neural network approach is not also used for head recognition… The reason is the current network needs a glyph to work upon, and there is yet no way to isolate a head glyph *reliably*. [1]

Music fonts hierarchy

Not all symbols are available in every music font family.

To cope with this situation, Audiveris has defined a hierarchy between music font families. Hence, when a given symbol is not available in a family, the families above in hierarchy are transitively searched until the symbol is found.

The current family hierarchy is as follows:[2]

```
Bravura
├── Leland
├── Primus
├── Finale Jazz
|    └── Jazz Perc
└── Musical Symbols
```

Text fonts MODIFIED IN 5.6

Texts recognition is less sensitive to font family, compared to music symbols recognition.
The main reason is that Audiveris delegates texts recognition to Tesseract OCR.[3]

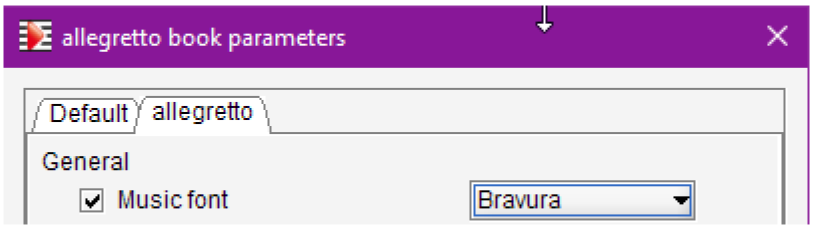
In the 5.3 release, the user could tell Audiveris which text font family to use among the families available (**Sans Serif** (the default), **Serif** and **Finale Jazz Text**):

Since the 5.6 release, Audiveris uses the OCR’d word attributes to decide which font name and style best fit any given word. And the user can manually modify the result.

Please refer to the use of word attributes in the section on word editing.

Selection of music font family

In interactive mode, we can use the `Book → Set book parameters` dialog in its `Music font` field:



☑ Music font Bravura

In batch mode, we can make the music choice on the command line interface.

The choice is between *Bravura*, *Leland*, *Primus*, *FinaleJazz* and *JazzPerc*:

```
-constant org.audiveris.omr.ui.symbol.MusicFont.defaultMusicFamily=JazzPerc
```

Footnotes

[1] There are on-going works on global recognition without prior glyph segmentation. See this Audiveris Wiki article in its “6.x prototype” section. ↩

[2] The old “Musical Symbols” family has been totally replaced by “Bravura” family. It is kept only for samples generation from old `.omr` projects, something the end user can safely ignore. ↩

[3] In some cases, we have observed rather poor OCR results with a Jazz-like text font, perhaps because lowercase characters are displayed as small uppercase characters. To be further investigated. ↩

Logical Parts SINCE 5.3

Mapping the individual parts of every system in every page to logical parts defined at score level is a topic where we may have to help the OMR engine.

TABLE OF CONTENTS

Definition

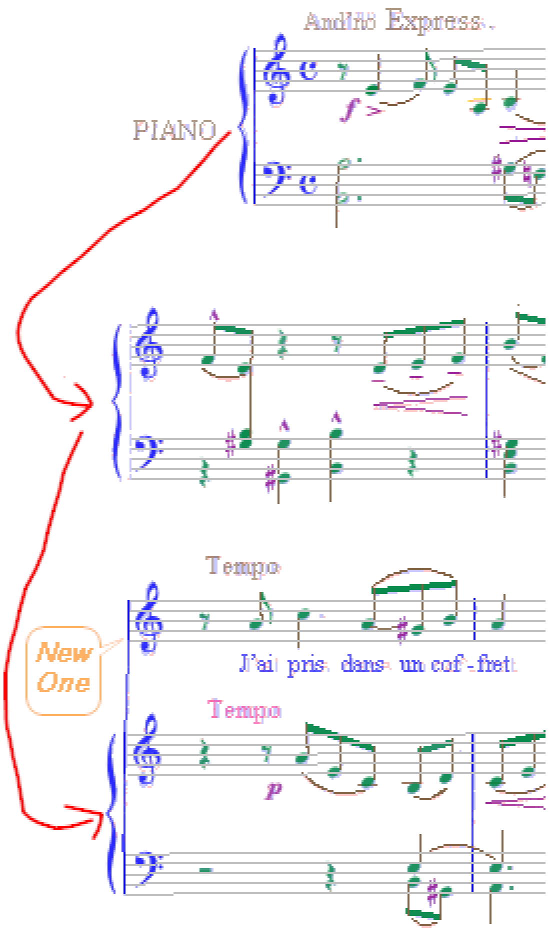
A part is generally meant to represent a given music instrument. It contains one stave, or two staves (piano example), or even 3 staves (organ).
We refer to these parts as “*physical*” parts.

A system is a vertical collection of parts, to be played in sync.
The systems, all composed of (physical) parts, are played one after the other.

In many cases, one (physical) part in a system corresponds to another (physical) part in the following system(s). We refer to this logical sequence as a **logical** part, which represents the same instrument all along the score.

Parts mapping

In some cases, from one system to the next, the system composition in terms of physical parts may vary as shown in the following example:

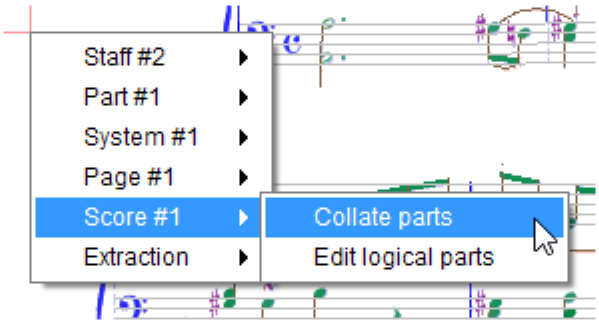


- System #1: A part made of 2 staves. This part is named “PIANO” in its left margin.
- System #2: A part made of 2 staves. No name is indicated but we can easily assume it’s still the piano instrument.
- System #3: 2 parts:
    - A 1-staff part, with no name. The singer joining the party?
    - A 2-staff part, with no name. Still the piano.

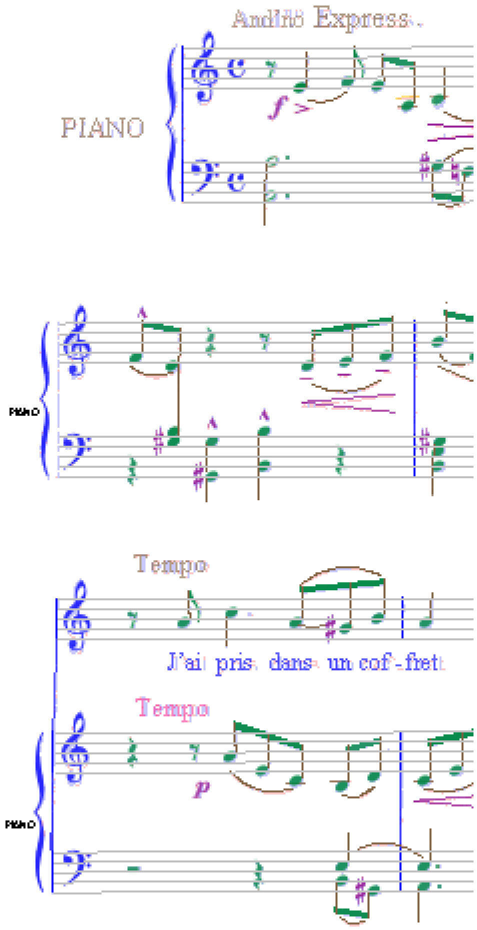
So we come up with 2 logical parts: a (no name) singer part and a PIANO part.

The OMR engine is able to determine this easy mapping on its own, by collecting and organizing each system part, one after the other.
We call this action: “***Part collation***” (into logical parts).

Part collation is automatically triggered at the end of score transcription, and before any export to MusicXML.
We can also manually trigger parts collation. This is done via a right-click in a score area, which opens a contextual menu as follows:



To visualize the current mapping, one easy way is to use the `View → Show part names` pull-down menu. This results in:

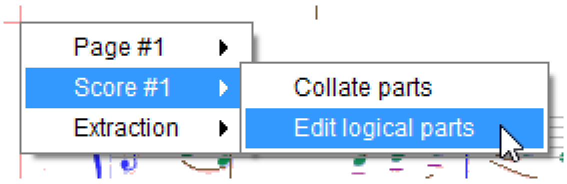


Each physical part is prefixed with the name of its corresponding logical part.

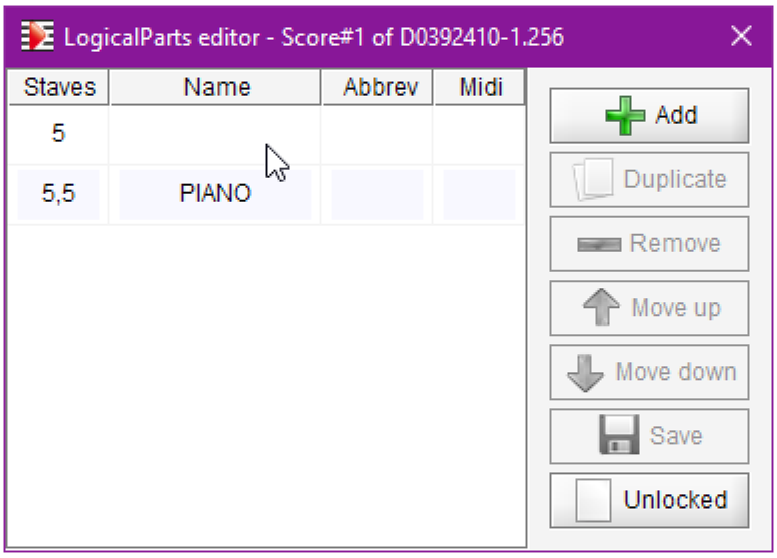
- System #1
  - PIANO (the existing “PIANO” text is displayed)
- System #2
  - PIANO (displayed in small size with a slight vertical offset)[1]
- System #3
  - (blank) since we have yet no name for this logical part
  - PIANO (displayed in small size with a slight vertical offset)

Editing of logical parts

Still via a right-click in the score area, we can open an editor on logical parts:



In our example, this editor will open as follows:



LogicalParts editor - Score#1 of D0392410-1.256

Staves	Name	Abbrev	Midi
5			
5,5	PIANO		

On the left side, we have a sequence of rows, one per logical part:

- **Staves**: The physical configuration of staves in part:
  - A comma-separated sequence of numbers, one per staff in part, giving the staff line count.
  - If the sheet contains two populations of staff height, the character **’s’** (for **s**mall) is appended when describing a **small** staff (for example: 5s).
- **Name**: The (logical) part name, or blank
- **Abbrev**: The related abbreviation if any
- **Midi**: The specified MIDI channel ~~(left blank, reserved for future use)~~ See the next section on MIDI assignments.

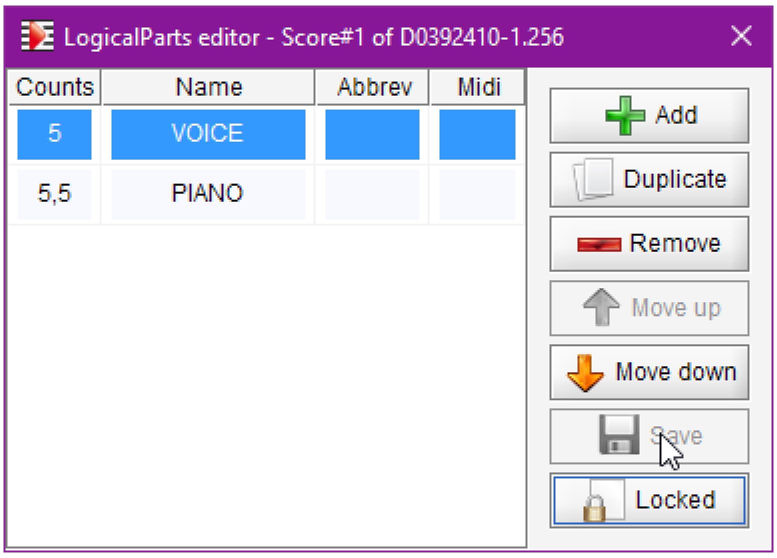
The picture above presents the content of the logical parts editor, automatically pre-populated by parts collation on our example.
We can see that 2 logical parts have been detected:

1. Part #1 contains just one staff, of standard size, composed of 5 staff lines.
   No name has been detected for this part.
2. Part #2 is a sequence of 2 staves, each composed of 5 staff lines.
   The string “PIANO” has been detected as the part name.

All these fields are editable.

On the right side, we have a collection of buttons to add, duplicate or remove parts, to move them up or down, and to save the logicals configuration.

For instance, we can decide to name the 1-staff part: “VOICE” to be consistent with the existing “PIANO” name. That’s all we need to do, so we save it:

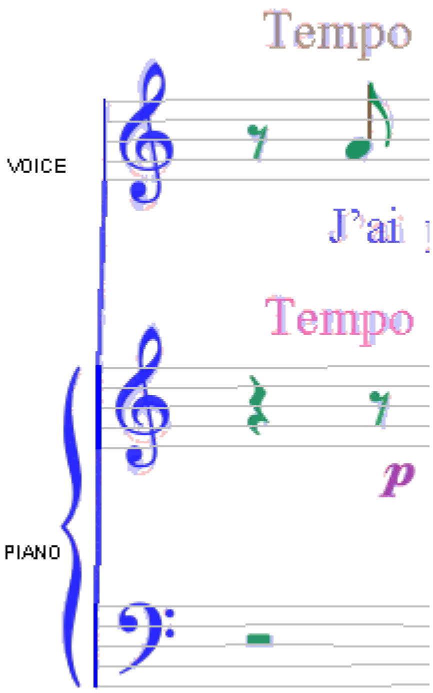


LogicalParts editor - Score#1 of D0392410-1.256

Counts	Name	Abbrev	Midi
5	VOICE		
5,5	PIANO		

Notice that the lower-right button (which was “*Unlocked*” in gray) is now “*Locked*” in black.
This means that the logicals just saved are now locked (but can be manually unlocked if so desired).

Notice also that the main score window now displays the updated parts names:



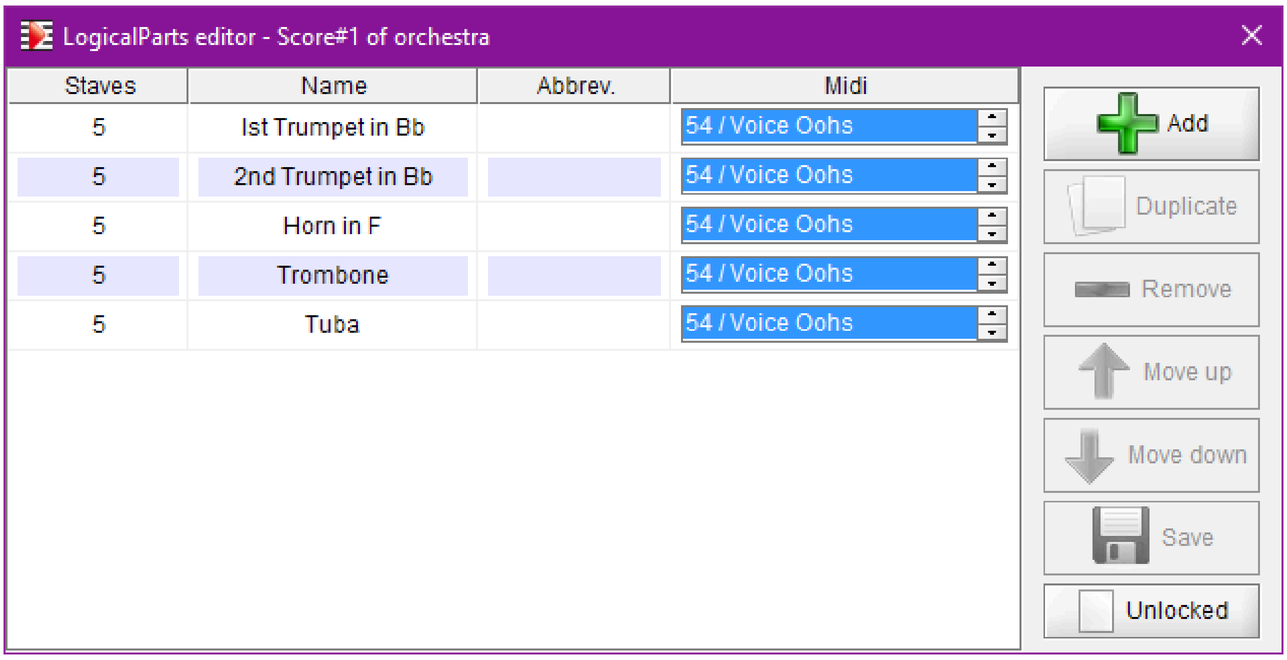
MIDI assignments

Before the 5.6 release, the MIDI program for a logical part was automatically assigned, and exported to MusicXML, based on the number of staves in the logical part:

- For a 1-staff part: **voice**
- For a 2-staff part: **piano**
- For a 3-staff part: **organ**

Starting with the 5.6 release, via the logical parts editor, we can manually assign the MIDI program of each logical part.

In the picture below, we can see an editor with 5 parts. Each part is composed of one staff and is, by default, assigned the “voice” instrument (program number: 54).



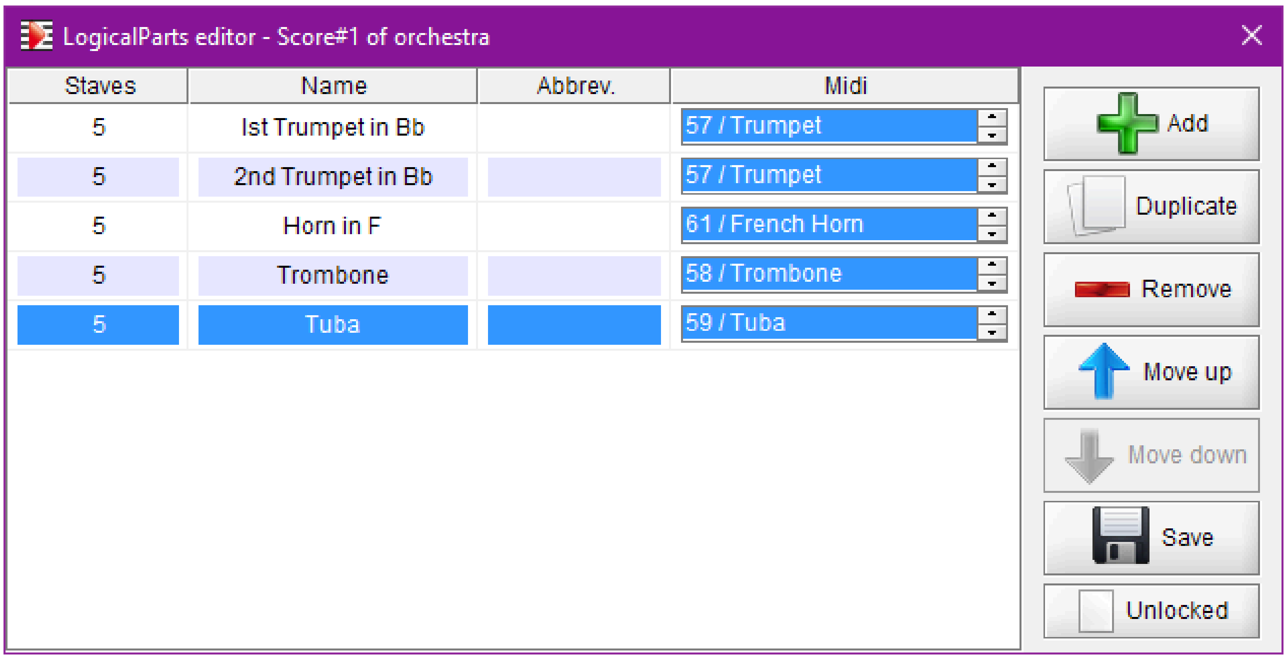
Staves	Name	Abbrev.	Midi
5	Ist Trumpet in Bb		54 / Voice Oohs
5	2nd Trumpet in Bb		54 / Voice Oohs
5	Horn in F		54 / Voice Oohs
5	Trombone		54 / Voice Oohs
5	Tuba		54 / Voice Oohs

This is not satisfactory, especially when compared to the OCR’d names of the parts… Perhaps one day the OMR engine will be able to infer the MIDI program from these OCR’d part names.

As a first step, the user has the ability to manually modify these assignments, via the combo boxes in the MIDI column:

- Either by using the up/down arrows to browse through the MIDI list of 128 instrument names, ranging from “*1 / Acoustic Grand Piano*” to “*128 / Gunshot*”, and then click on the chosen name,
- Or by directly typing the program numeric value.

We finally get these MIDI assignments:



Staves	Name	Abbrev.	Midi
5	Ist Trumpet in Bb		57 / Trumpet
5	2nd Trumpet in Bb		57 / Trumpet
5	Horn in F		61 / French Horn
5	Trombone		58 / Trombone
5	Tuba		59 / Tuba

Unlocked/Locked logicals configuration

The “Parts collation” action works differently according to the unlocked/locked status of our logicals configuration.

In the **Unlocked** status, the logicals configuration is (re-)written from scratch.
For each physical part found in sequence in the score:

1. The physical part name and characteristics are collected,
2. A compatible logical part is either reused or created,
3. The physical part is mapped to its logical part.

In the **Locked** status, the logicals configuration is in read-only mode.
For each physical part – not manually mapped – found in sequence in the score:

1. The physical part name and characteristics are collected,
2. The physical part is assigned to a compatible logical part or a warning is displayed for this un-mappable physical part.

Mapping algorithm

The parts mapping works as follows.
For every physical part checked against a logical part:

- Parts physical configurations must match (number of staves, staff-line counts, staff small attribute if any).

- If a physical part name has been recognized or assigned, it is compared, ignoring case, with logical name or logical abbreviation.
- If physical and logical do not match, check is made against the next logical if any.

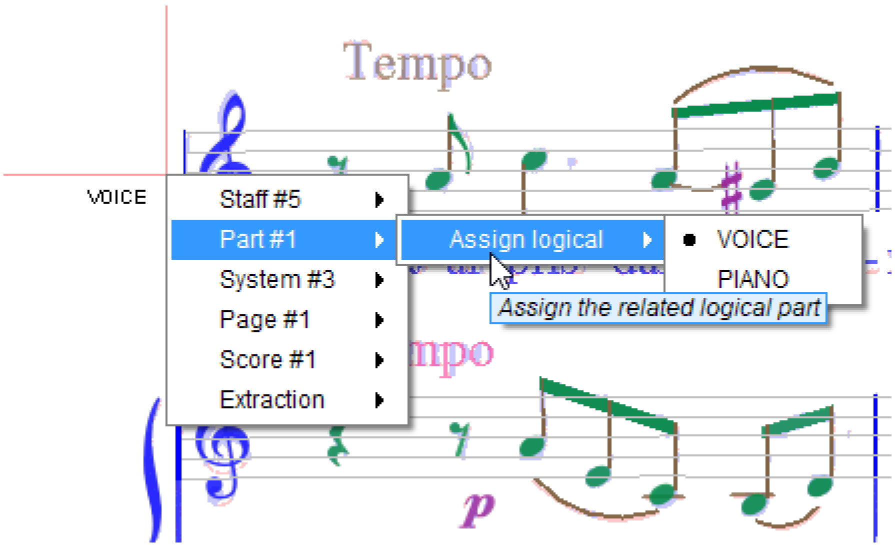
Within every system, this part checking is performed as follows:

- If the system contains a standard 2-staff part (typically the piano part), then checking goes bottom-up from the piano part, and goes top-down for the parts located below the piano part.
- If there is no piano part, checking is done bottom-up.

Manual mapping

If the score gets too complex for the OMR engine, we can always fall back to the manual mapping of any “un-mappable” physical part.

This is done via a right-click in the physical part to be manually mapped:



We simply have to pick up the corresponding logical among the presented sequence of logicals.

See a real-life example in Can’t Take My Eyes Off of You

Footnotes

[1] The created part names are displayed slightly off of the expected vertical location, to avoid hiding any potential existing part name. ↩

How-to guides / Specific features / Octave Shift

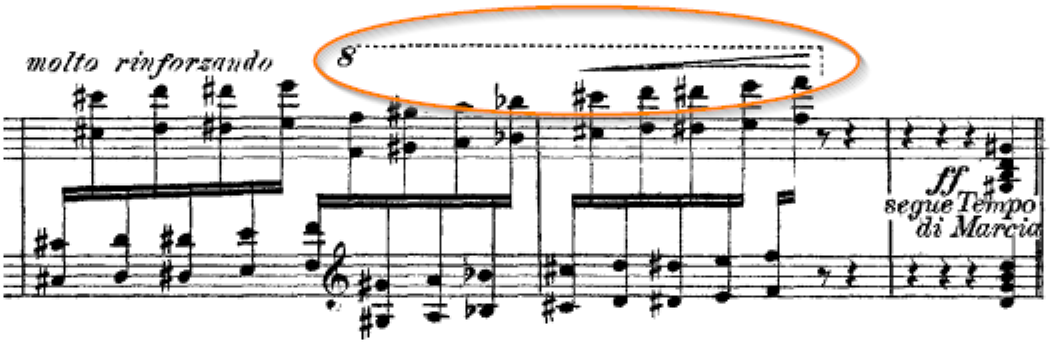
Octave Shift SINCE 5.3

An octave shift indicates that some notes of a staff should be played one or several octaves higher or lower than actually printed.

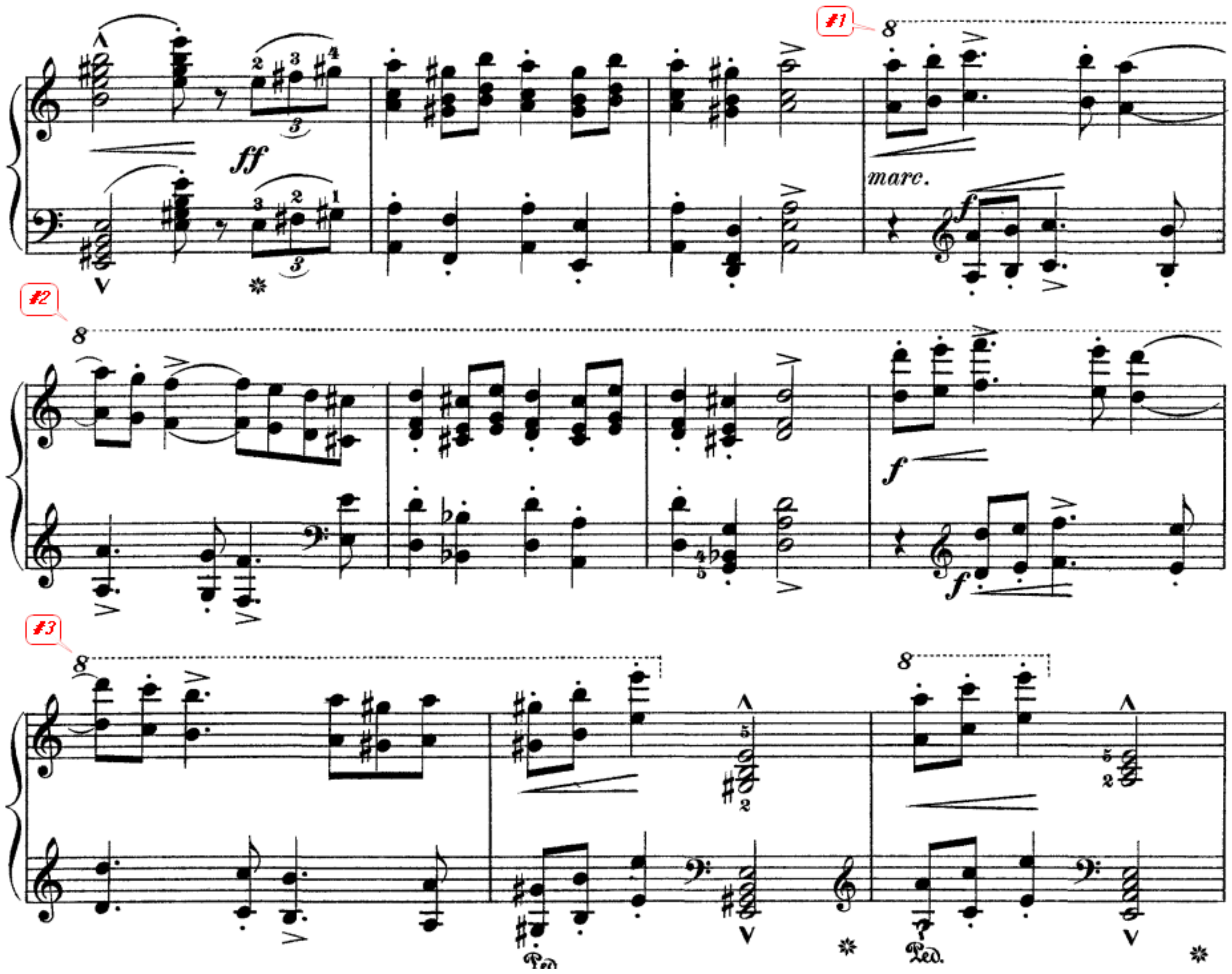
TABLE OF CONTENTS

Examples

Here is an example of a one-line shift:



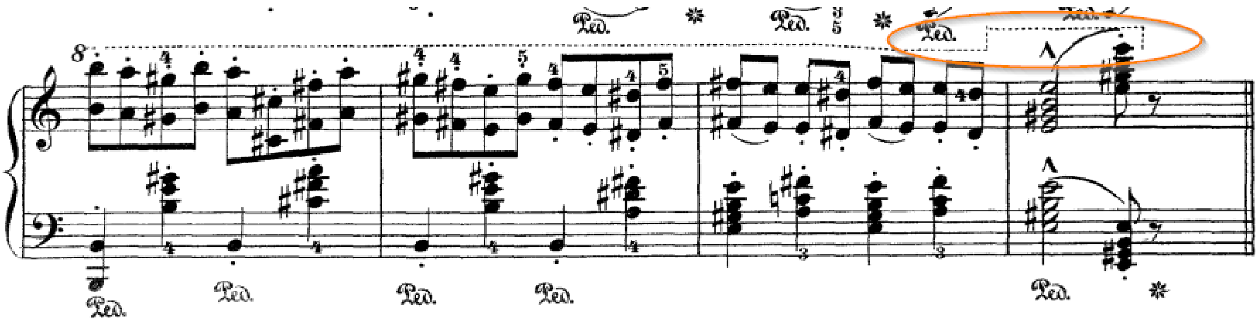
And here is an example of a multi-line shift:
We can consider we have one “logical” shift composed of three “physical” shifts.



Recognition

Automatic recognition of octave shifts by the OMR engine is difficult for various reasons:

- The value is not always just a number to be found in {8, 17, 22} set. There can be a suffix, such as “va” or “vb”, sometimes located slightly above or below the number.
  Sometimes but not always, “va” and “vb” stand respectively for “ottava alta” and “ottava bassa”.
- The line is often composed of dots or short dashes, that may have been consumed by the OCR TEXT step.
- Even if they survive OCR, these long sequences of dots are difficult to recognize in the SYMBOLS step by the current classifier, which requires identified glyphs.
- The line itself is sometimes far from being a straight line, see the image below.

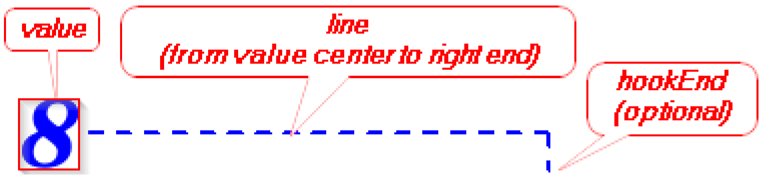


Model

Audiveris has chosen to focus on a simplified model.

An octave shift element is defined as a horizontal sequence of:

1. A number value (8, 15 or 22),
2. A horizontal dashed line,
3. An optional vertical ending hook pointing up or down to the related staff. The ending hook appears only at the end of the last (physical) shift of a perhaps longer logical shift.



The vertical location of this octave shift element with respect to the related staff indicates its kind (`ALTA` if above the related staff, `BASSA` if below).

A multi-line (logical) shift is implemented as a vertical sequence of several (physical) shifts.

The OMR engine may recognize just the value portion of an octave shift, but it is then rather easy for the end-user to edit the line portion.

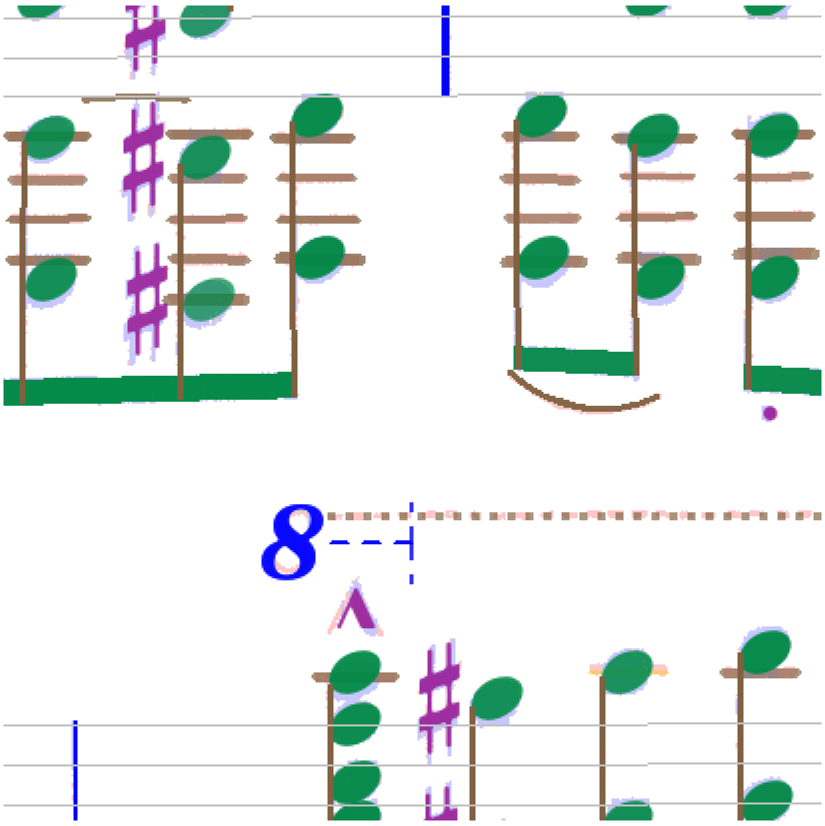
Editing

Location

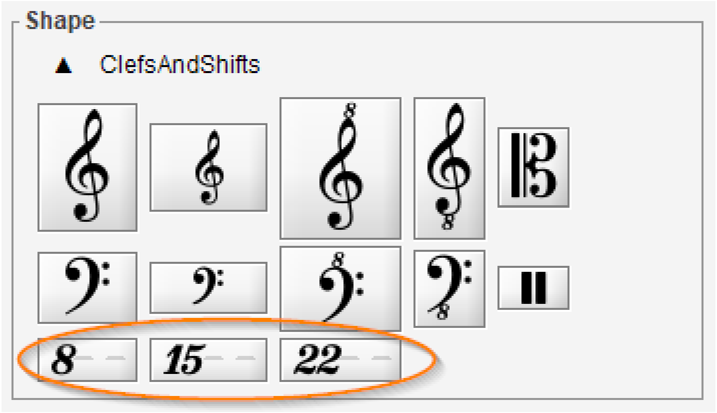
First things first, we need the shift value.

- The shift can be detected by the OMR engine, or manually assigned on some underlying glyph. The closest staff is by default the related staff. When the glyph is located between two systems, it can lead to two mutually

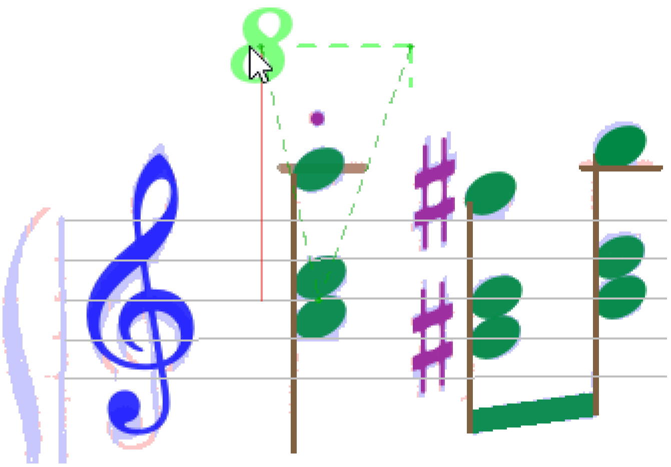
exclusive Inters, one for the system above and one for the system below, as shown by the picture below (notice the two end hooks).
In that case we can manually discard the `Inter` we don’t want to keep.



- The shift can also be dragged from the shape palette and dropped at proper location.

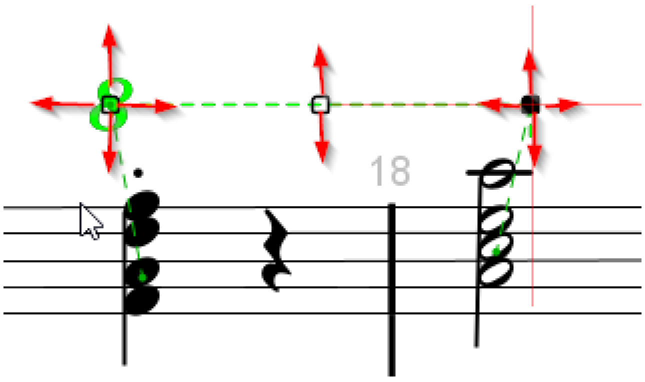


As we hover on staves, the shift related staff changes and so does its ending hook. At drop time, the last staff is kept as the related staff.



Single line editing

As usual, a double-click on the shift `Inter` opens a dedicated editor on it.



For a one-line shift, there are 3 handles:

- Left handle can move horizontally and vertically
- Middle handle can move only vertically
- Right handle can move horizontally and vertically.

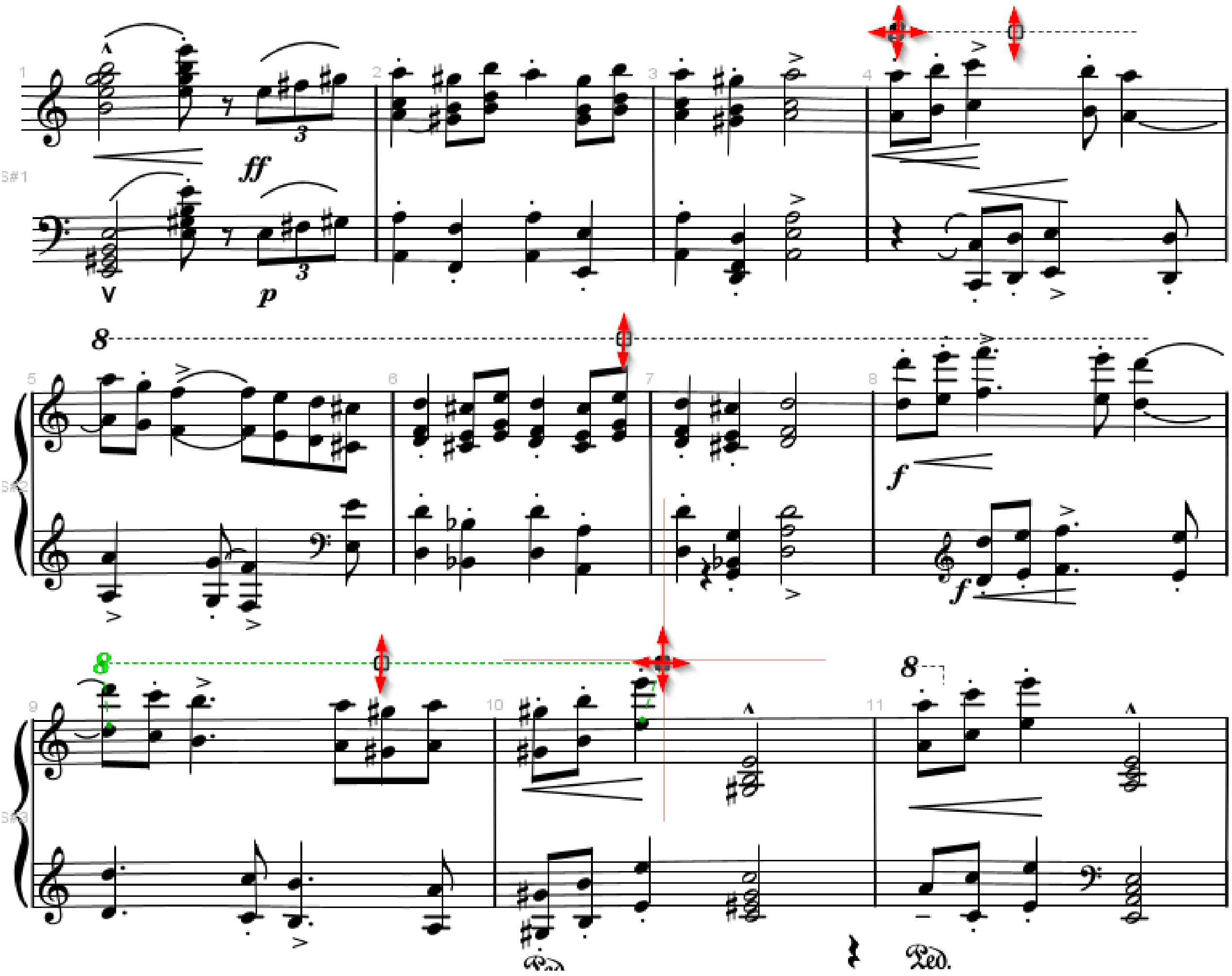
Dragging a handle horizontally allows to extend or shrink the shift line.

Dragging a handle vertically is more complex:

- When slightly dragged vertically, the whole shift is translated but constrained to stay below the upper staff if any as well as above the lower staff if any.
- If we force the left handle way beyond the upper staff until the *corresponding* staff in the system above, the initial shift gets separated in two shifts, one for each staff.
- A similar mechanism applies for the right handle when dragged way beyond the lower staff.

In these forced cases, the initial one-line shift has evolved to a multiple-line shift. And it can continue its evolution.

Multiple line editing



The picture above represents the current status of a 3-line shift being edited:

- The first line goes until the staff right side;
  It has a left handle and a middle handle
- Any middle line spans the whole staff width;
  It has just a middle handle
- The last line starts from staff left side;
  It has a middle handle and a right handle

Forcing the first line downwards (or the last line upwards) allows to shrink the shift and reduce the number of lines, potentially down to a single line shift.

End

Clicking outside any handle completes the current editing.

Selecting and deleting any line of a multiple-line shift deletes the whole shift.

Editing can be undone and redone.

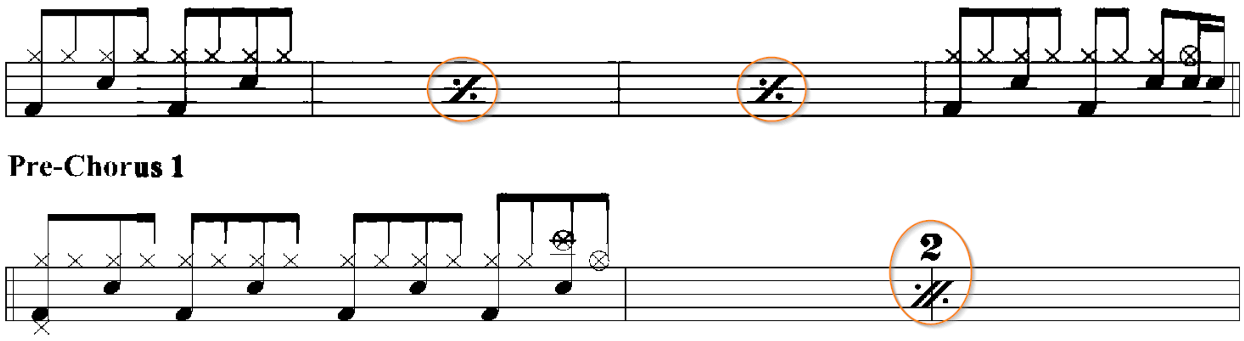
How-to guides / Specific features / Bar Repeat

Bar Repeat SINCE 5.3

A “bar repeat” sign indicates that preceding written measures must be repeated exactly.

TABLE OF CONTENTS

Example



In the example above, we have two systems, the first one with 4 measures #1, #2, #3 and #4
and the second one with 3 measures #5, #6 and #7:

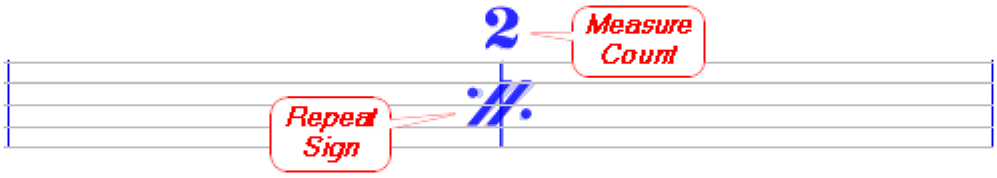
- On the first system, the measures #2 and #3 contain both a one-bar repeat sign, to repeat the measure #1 twice.
- On the second system, a two-bar repeat sign appears right above the barline between the measures #6 and #7, to repeat the previous two measures (#4 and #5).

We could also encounter a four-bar sign, to repeat the four previous measures.

Model

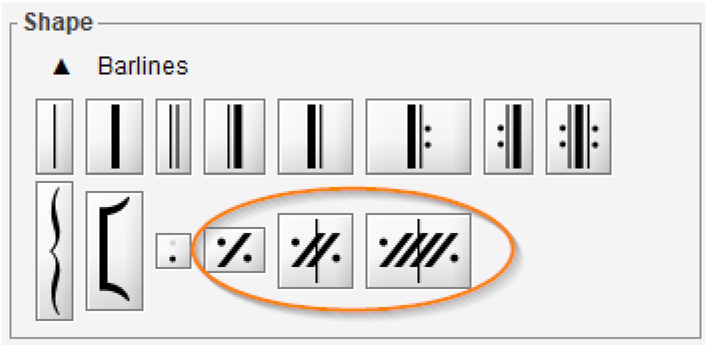
Since release 5.3, Audiveris OMR engine is able to recognize both:

- The repeat sign itself, with either 1, 2 or 4 slashes.
- An optional measure count that can appear above the sign. It should correspond to the count of slashes in the repeat sign.

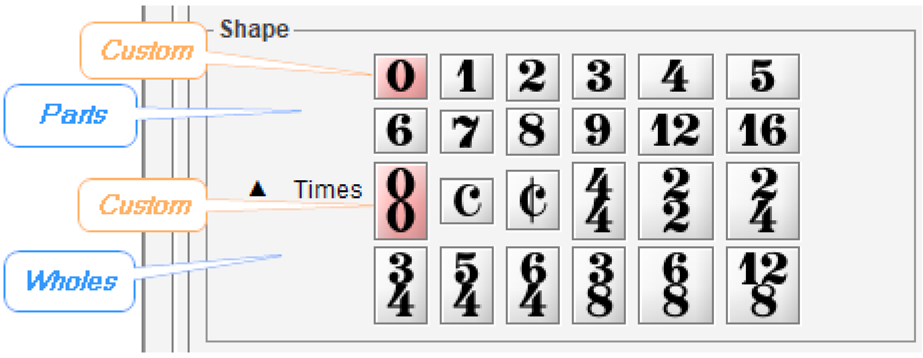


Editing

If needed, we can manually assign or drag these signs from the shape palette, using the three symbols available in the `Barlines` set.



The measure count is not mandatory, but can be manually assigned or dragged from the shape palette, in the `Times` set.



We can choose a predefined item from the `Parts` numbers.

Or pick up the custom `0` item and then manually assign the desired value.

Output

The MusicXML output is correctly populated with these bar repeat signs and with the content of the repeated measures.

Regarding the display:

- When read with Finale (version 27), the repeat signs are displayed as expected.
- When read with MuseScore (version 3.6.2) the repeat signs are not displayed but the repeated measures are shown as if they had been physically duplicated.

Regarding the playback:

- In both cases (Finale and MuseScore), the repeated measures sound as expected.

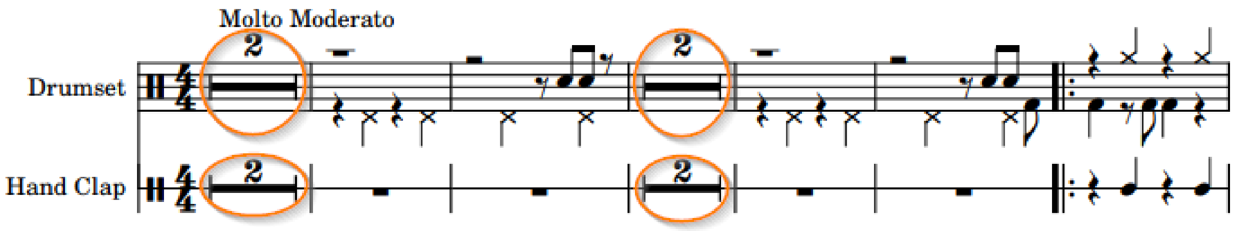
Multi-Measure Rest SINCE 5.3

A multiple measure rest indicates a rest of more than one measure.

TABLE OF CONTENTS

Example

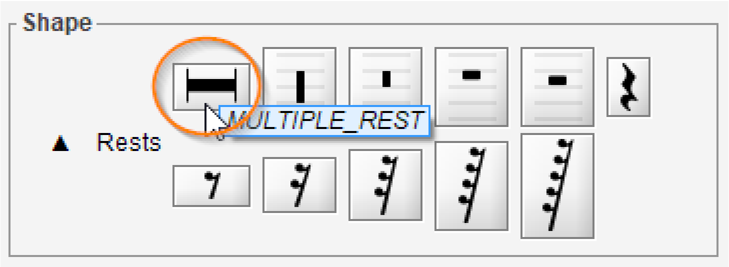
It is drawn as a thick horizontal line centered on staff middle line with serifs at both ends, completed by a time-like number drawn above the staff.



In the example above, we have instances of multiple measure rests, on 5-line and 1-line staves.

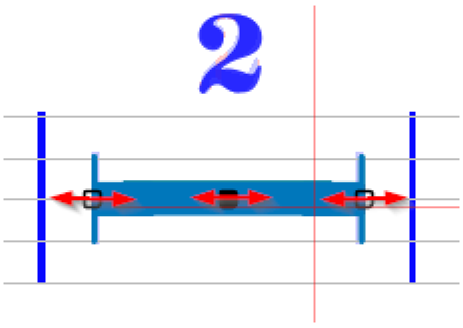
Editing

If not detected, we can still assign or drag a multiple rest item from the shape palette in its `Rests` set:



A double-click on a multiple rest enters its fairly limited editing status:

- Vertically, the item remains stuck to the staff mid-line,
- Horizontally, the mid handle can translate the whole item, while a side handle can resize it.

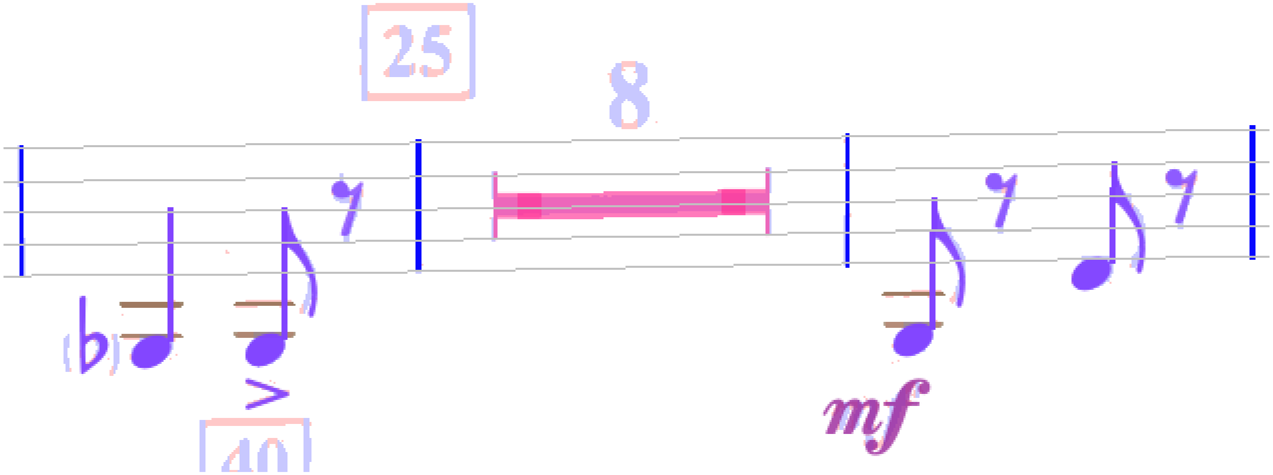


And, as for bar repeats editing, the measure count item can be assigned or dragged from the shape palette in its `Times` set.

Model

Since release 5.3, with no need for a specific processing switch, the multiple rests should be detected (during the `BEAMS` step), together with the related measure count number (during the `SYMBOLS` step).

NOTA: If the engine cannot recognize the measure count number, the multiple rest is set in the “*abnormal*” status. It is consequently displayed in red, as in the picture below, where the symbol “8” was not recognized:



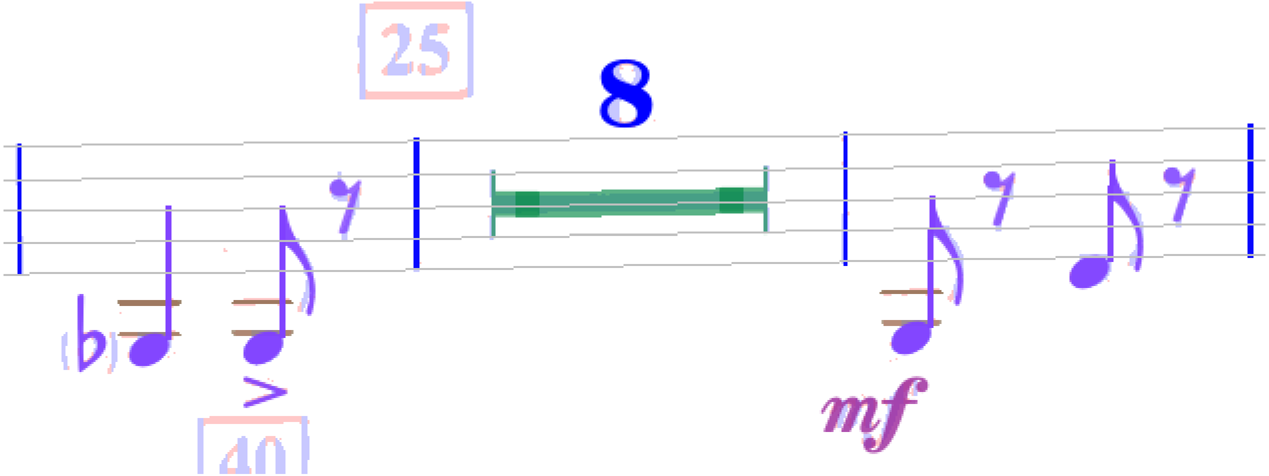
And during the reduction performed by the following step (`LINKS`), all *abnormal* inters are deleted, including the multiple rest shown above!

To remedy this, starting with version 5.8, if no measure count is recognized or provided manually:

1. The multiple rest is **not** deleted
2. It is still exported in MusicXML format, but with a **default count** of 1, and a warning is issued to the user, like:

   > Measure{#25} Multirest with no measure count, using default value 1

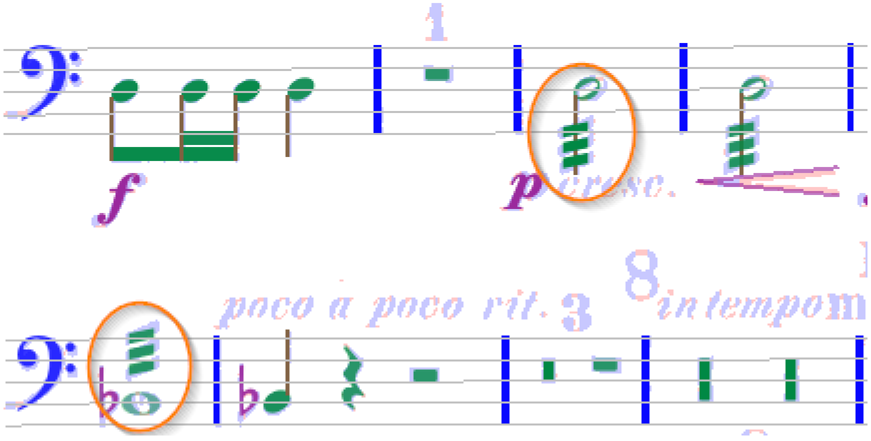
We can always manually fix that by selecting the “8” glyph and assigning it the shape 8 (TIME_HEIGHT), either via the popup Glyphs menu, or via the Times shapes palette:



Note the multiple rest is no longer displayed in red, and it will be exported in MusicXML with the correct count of 8 measures.

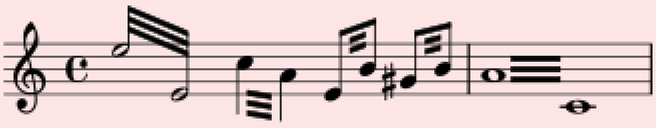
Tremolo SINCE 5.3

Since release 5.3, Audiveris can handle tremolos as depicted below:

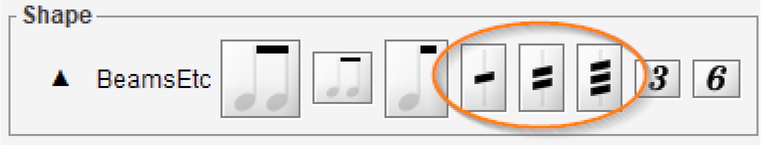


A tremolo is composed of 1, 2 or 3 slanted segments.

It must be horizontally aligned with a chord, either a stem-based chord or a stem-less chord. Both cases are shown in picture above.

> **WARNING**
>
> Audiveris does not yet address the case of tremolos located ***between*** two chords, as shown in the picture below, stolen from Wikipedia:
>
> 

If the OMR engine has not recognized a tremolo, we can always assign or drag the desired tremolo from the shape palette, using the `BeamsEtc` shape set.



Tremolo editing is limited to translations of the item, which will snap on compatible chords nearby.

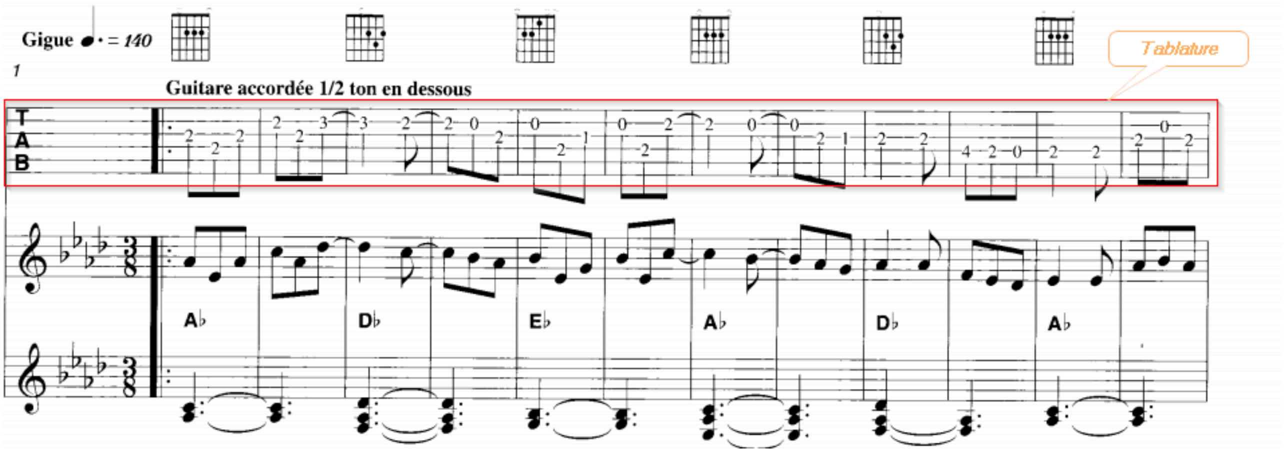
How-to guides / Specific features / Tablature

Tablature SINCE 5.3

A *tablature* is a special kind of staff that represents the physical locations of strings and frets to be pressed.

TABLE OF CONTENTS

Example



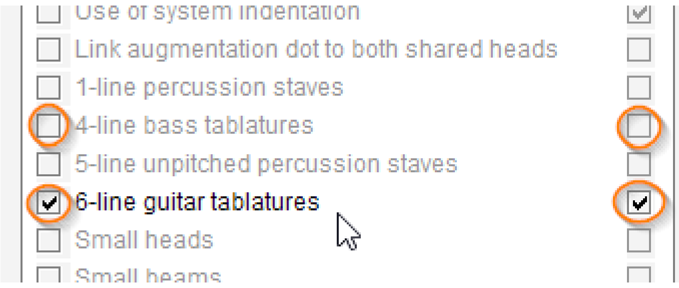
This tablature example contains 6 lines, and corresponds to a 6-string guitar. There exist also tablatures for 4-string bass guitar, represented with 4 lines.

While a standard staff would begin with a clef (G, F or C), this tablature staff begins with a “TAB” indication.

Detection

By default, tablatures are not detected.
To enable their detection, we have to explicitly set the processing switches

(either or both the 4-line or 6-line tablatures as depicted below):



□	Use of system indentation	☑
□	Link augmentation dot to both shared heads	□
□	1-line percussion staves	□
□	4-line bass tablatures	□
□	5-line unpitched percussion staves	□
☑	6-line guitar tablatures	☑
□	Small heads	□
□	Small beams	□

The GRID step of OMR engine detects clusters of regularly spaced horizontal lines.
Typically a cluster of 5 lines is considered as a “standard” 5-line staff.

If properly enabled, 4- or 6-line clusters are detected as “tablatures” staves.

Processing

There is no processing *per se* by the OMR engine.

The tablature is merely identified as such and its staff area is carefully ignored by any subsequent processing.

There is thus no risk of false detection of notes, slurs, text, symbols, etc in this area.

> The ignored area is defined by the tablature staff augmented by a slight vertical margin (this area is indicated by a red rectangle in the example).
> But there is no way yet to ignore the pixels from stems or beams located **outside** the tablature.
> Perhaps a broader solution would be to let the user interactively **blacklist** certain image areas, starting with properly defined tablature areas. In the example shown, it could also apply to these guitar chord diagrams that appear just above the tablature.

Fingering and Plucking COMPLETED IN 5.3

These are technical informations for guitar or similar instrument, to indicate how a note should be played.

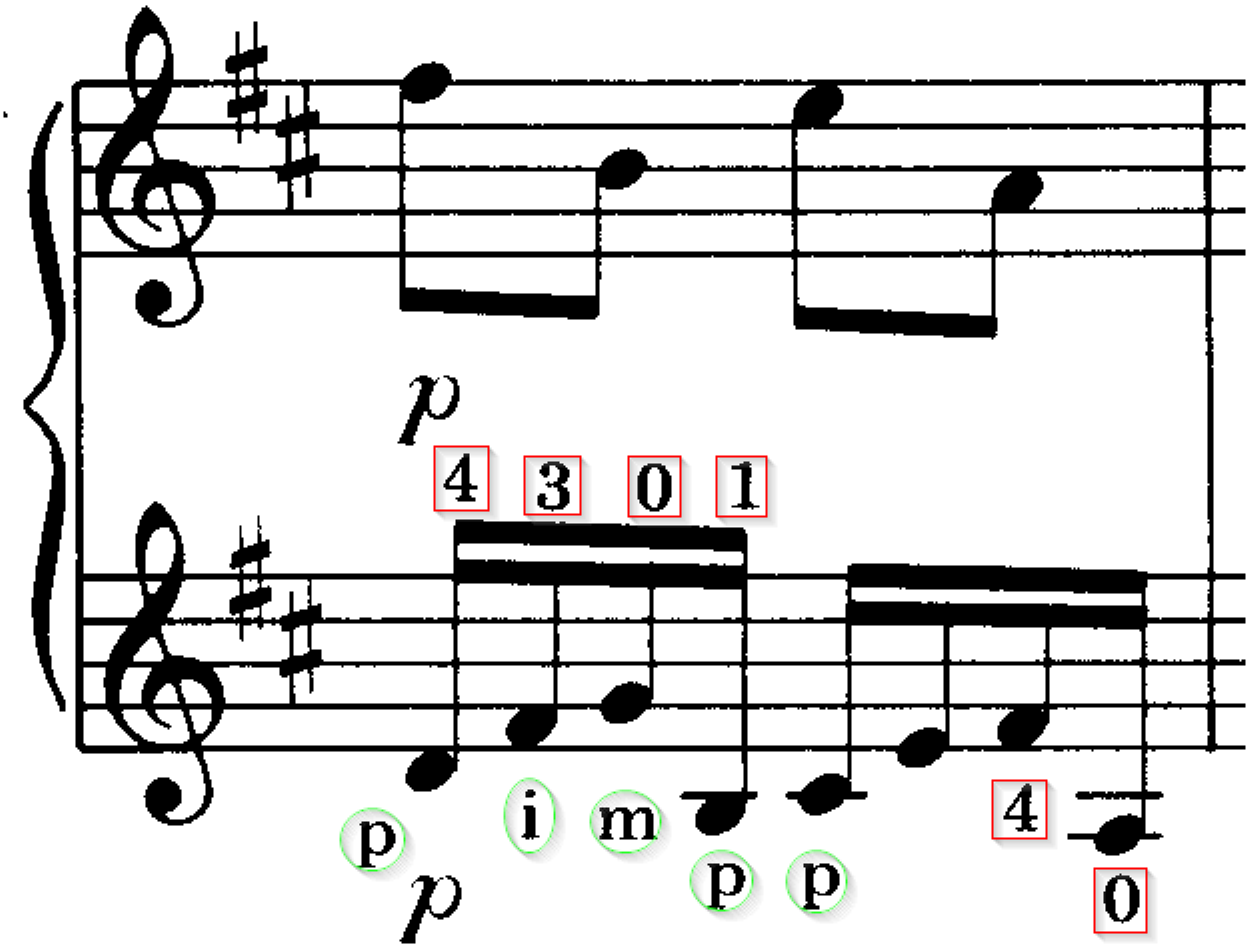
[Assuming a right-handed person]:

- **Fingering** describes the left-hand finger, via a digit number (0, 1, 2, 3, 4)
- **Plucking** describes the right-hand finger, via a letter (`p`, `i`, `m`, `a`)

TABLE OF CONTENTS

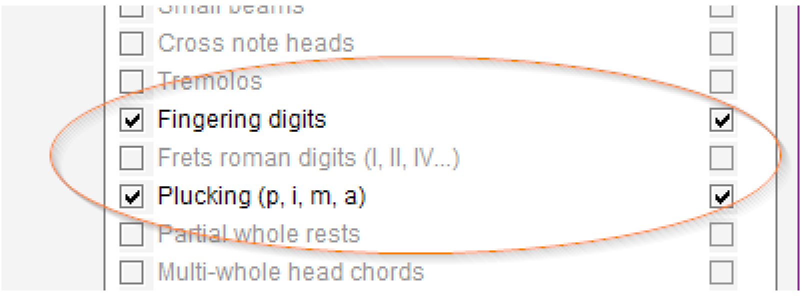
Example

Here is an example with fingering indications (in red squares) and plucking indications (in green circles):



Detection

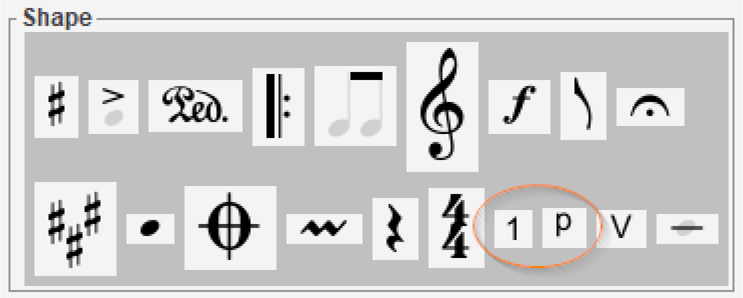
The OMR engine must be explicitly told to detect these indications. This can be done via the processing switches in the `Book → Set book parameters` pull-down menu:

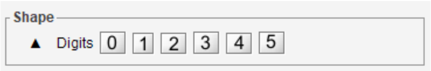
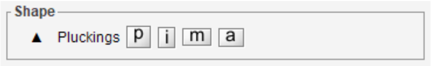


□ Small beams □
□ Cross note heads □
□ Tremolos □
☑ Fingering digits ☑
□ Frets roman digits (I, II, IV...) □
☑ Plucking (p, i, m, a) ☑
□ Partial whole rests □
□ Multi-whole head chords □

Insertion

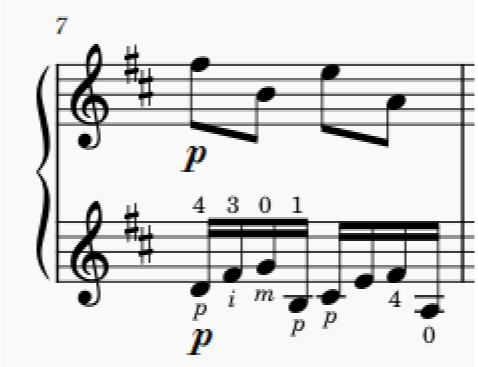
Regardless of the switches values, we can always manually assign or drag & drop these indications from the shape board:



Shape Set	Palette
Fingerings	 Shape ▲ Digits 0 1 2 3 4 5
Pluckings	 Shape ▲ Pluckings p i m a

Export

These indications get exported to MusicXML and are thus visible from MuseScore for example:



Advanced features

Audiveris provides a few features that are mostly interesting only for the advanced user. They are all gathered in this chapter.

The `Samples` and the `Training` chapters are specifically intended for users who need to modify the behavior of the neural network underlying the glyph classifier.

TABLE OF CONTENTS

Command line interface

TABLE OF CONTENTS

Syntax

Any argument beginning with the `@` character is considered as the name of a text file to be immediately expended *in situ* (the text file is assumed to contain one argument per line).

CLI syntax is displayed as follows when the `-help` argument is present:

```
Audiveris Version:
   5.4

Syntax:
    audiveris [OPTIONS] [--] [INPUT_FILES]

@file:
    Content of file to be extended in line

Options:
 -help                                            : Display general help then stop
 -batch                                           : Run with no graphic user interface
 -sheets int[]                                    : Select sheet numbers and ranges (1 4-5)
 -transcribe                                      : Transcribe whole book
 -step [LOAD | BINARY | SCALE | GRID | HEADERS |  : Define a specific target step
 STEM_SEEDS | BEAMS | LEDGERS | HEADS | STEMS |
 REDUCTION | CUE_BEAMS | TEXTS | MEASURES |
 CHORDS | CURVES | SYMBOLS | LINKS | RHYTHMS |
 PAGE]
 -force                                           : Force step/transcribe re-processing
 -output <output-folder>                          : Define base output folder
 -playlist <file.xml>                             : Build a compound book from playlist
 -export                                          : Export MusicXML
 -print                                           : Print out book
 -constant key=value                              : Define an application constant
 -upgrade                                         : Upgrade whole book file
 -save                                            : In batch, save book on every successful step
 -swap                                            : Swap out every sheet after its processing
 -run <qualified-class-name>                      : (advanced) Run provided class on valid sheets
 -sample                                          : (advanced) Sample all book symbols
 -annotate                                        : (advanced) Annotate book symbols

Input file extensions:
    .omr        : book file  (input/output)
    [any other] : image file (input)

Sheet steps are in order:
    LOAD       : Get the sheet gray picture
    BINARY     : Binarize the sheet gray picture
    SCALE      : Compute sheet line thickness, interline, beam thickness
```

```
     GRID       : Retrieve staff lines, barlines, systems & parts
     HEADERS    : Retrieve Clef-Key-Time systems headers
     STEM_SEEDS : Retrieve stem thickness & seeds for stems
     BEAMS      : Retrieve beams
     LEDGERS    : Retrieve ledgers
     HEADS      : Retrieve note heads
     STEMS      : Retrieve stems connected to heads & beams
     REDUCTION  : Reduce conflicts in heads, stems & beams
     CUE_BEAMS  : Retrieve cue beams
     TEXTS      : Call OCR on textual items
     MEASURES   : Retrieve raw measures from groups of barlines
     CHORDS     : Gather notes heads into chords
     CURVES     : Retrieve slurs, wedges & endings
     SYMBOLS    : Retrieve fixed-shape symbols
     LINKS      : Link and reduce symbols
     RHYTHMS    : Handle rhythms within measures
     PAGE       : Connect systems within page
```

Arguments

These are the standard arguments that are listed when the help option is used. They are presented here in alphabetical order.

-batch

Launches Audiveris without any Graphic User Interface.

-export

Exports each book music as a MusicXML file.

-force

Forces reprocessing even if target step has already been reached.
This option is effective only when a target step is specified (see the `-step` option) or the `-transcribe` option is present.

-help

Displays the arguments summary as printed above, then exits.

-constant KEY=VALUE

Specifies the value of one application constant: [1]

- KEY being the *fully qualified* name of the constant,
- VALUE being the value to assign.

This is the CLI equivalent of the GUI pull-down menu `Tools → Constants`.

-output DIRNAME

Defines the path to the target output folder, that is the precise folder where all output files (`.omr`, `.mxl`, etc) should be stored.

if this option is not present, a default output folder is chosen according to the policy described in Standard folders section.

-playlist FILE.XML

Loads the provided `.xml` file as a playlist.

If in batch mode, the loaded playlist is used to build a compound book according to the playlist content.

If in interactive mode, the loaded playlist is used only to populated and display a `Split and merge` dialog. The user can then review and/or edit the playlist and potentially launch the building of the compound book at a desired location.

-print

Exports each book music as a PDF file.

-save

Saves each book OMR data to its `.omr` project file as soon as a sheet step is processed successfully.

This option is effective only in `-batch` mode.

-sheets N M X-Y

Specifies the IDs of sheets to process.

IDs are specified as a space-separated sequence of numbers (a sheet ID starts at 1).
Also, the X-Y notation is accepted, to refer to all IDs between X and Y included.
Mind the fact that X-Y must be one argument, with no space around the `-`

character.

This option is meant to initially open the book on a specific sheet, or to restrict processing to some sheets. If no sheet IDs are specified, all (valid) sheets are concerned.

Sheet IDs apply to all books referenced on the command line.

-step STEPNAME

Specifies a sheet target step.

This target step will be reached on every sheet referenced from the command line. This means all valid sheets if no explicit sheet numbers are specified.

For any given sheet, if the target step has already been reached, no further processing is done.
However, if the `-force` option is present, this sheet will be reset to BINARY and then processed again to the target step.

-transcribe

Transcribes each book.

`--`

This argument (a double dash: “`--`”) is not a real argument *per se*, but merely a delimiter so that each following argument in the command line is taken as an input file path (even if this argument begins with a `-` character).

FILENAME

Path to one input file.

If the file name extension is `.omr`, the file is an Audiveris project file which will be used as input / output.

For any other extension, the file is considered as an image input file.

Advanced Arguments

These arguments are made available for the advanced user.

-annotate

For each book, populates a Zip archive with images and symbol annotations derived from book `Inter` instances.

These annotations are meant to populate a dataset for training future Audiveris 6.x new classifiers (Page and/or Patch).

-sample

Populates each book sample repository with samples derived from the book `Inter` instances.

A book-level repository can be later merged into the global Audiveris sample repository in order to prepare a training of Audiveris 5.x Glyph classifier.

-run CLASS_NAME

Runs the specified Java class on each valid sheet.

CLASS_NAME must be the fully qualified name of a Java class, which must extend the abstract class `org.audiveris.omr.step.RunClass` and override its process() method:

```
public abstract class RunClass
{
    protected Book book;
    protected SortedSet<Integer> sheetIds;
    /**
     * Creates a new {@code RunClass} object.
     *
     * @param book     the book to process
     * @param sheetIds specific sheet IDs if any
     */
    public RunClass (Book book,
                     SortedSet<Integer> sheetIds)
    {
        this.book = book;
        this.sheetIds = sheetIds;
    }
    /**
     * The processing to be done.
     */
    public abstract void process ();
}
```

Footnotes

[1] `-constant` is a better name than the old `-option`, but both names are supported. ↩

Scanning of paper scores

Audiveris doesn't support direct scanning because there is still no open-source solution for cross-platform access to scanning devices from Java projects. We will need to use an external scanning utility. Fortunately, all major scanner manufacturers offer such a utility. There is also a variety of freely available scanning software for all common operating systems.

When scanning for music recognition, we should pay attention to the following advices:

- Prefer grayscale images to black-and-white or color images. This will help avoid unwanted symbols distortion in the scanning software. Audiveris will binarize the image by itself using a specific adaptive algorithm.
- Choose an optimal image resolution according to the following criteria:
  - Too low resolution (below 200 DPI) may hide key details while too high (more than 500 DPI) quickly turns into a significant waste of CPU and memory resources without any advantage for the recognition.
  - As a rule of thumb, **300 DPI** is generally a good resolution for a paper of standard A4 or US Letter size. For scores that exhibit small symbols, we'll need to increase the image resolution to 400 DPI.
  - An even better rule is to scan the image so that the resulting vertical distance between two staff lines is about **20 pixels**.
- Paper placement should be made carefully to avoid image rotation, warping, shadows and dark vertical stripes. Audiveris usually detects any rotation and/or warping but, to keep image quality intact, it never attempts to graphically correct them. Instead, the distorted staff lines are kept as they are, and are used as local references for any symbol processing.
- Any post-processing of scanned images in the scanning software should be turned off. Special settings like “Line Art”, “Text”, “Dithering” or “Halftone” should be disabled because they often lead to unwanted image distortions. We should prefer a dedicated image processing software in order to carefully enhance low-quality images when needed. See the Improved Input section.

Improved Input

Sometimes the base image is of rather bad quality. Although the human’s eye can identify the notes correctly, the algorithms have problems to detect the components of a score correctly.

TABLE OF CONTENTS

Using Gimp

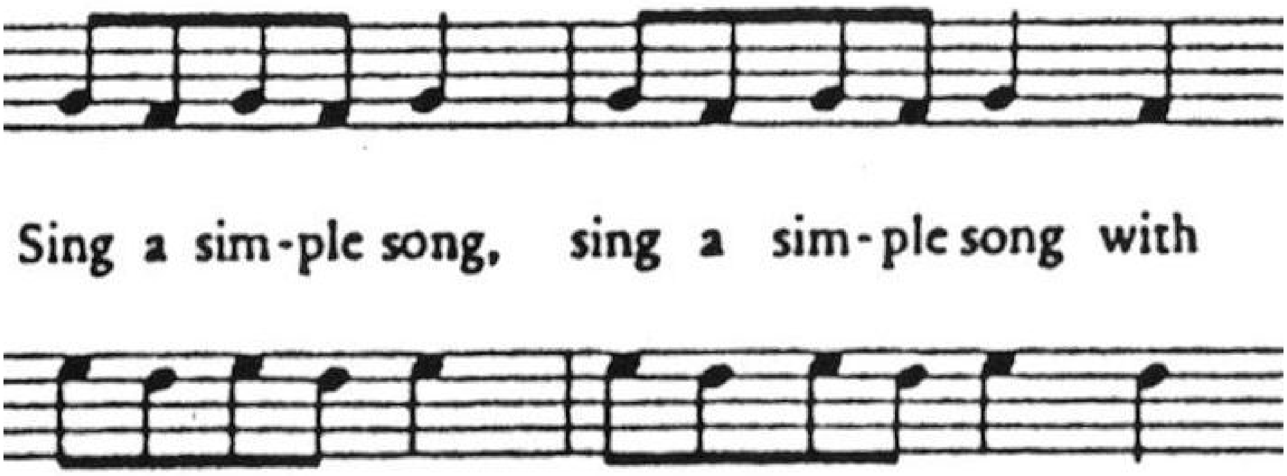
[Contribution by Baruch Hoffart]

Here are some possible improvements using Gimp.

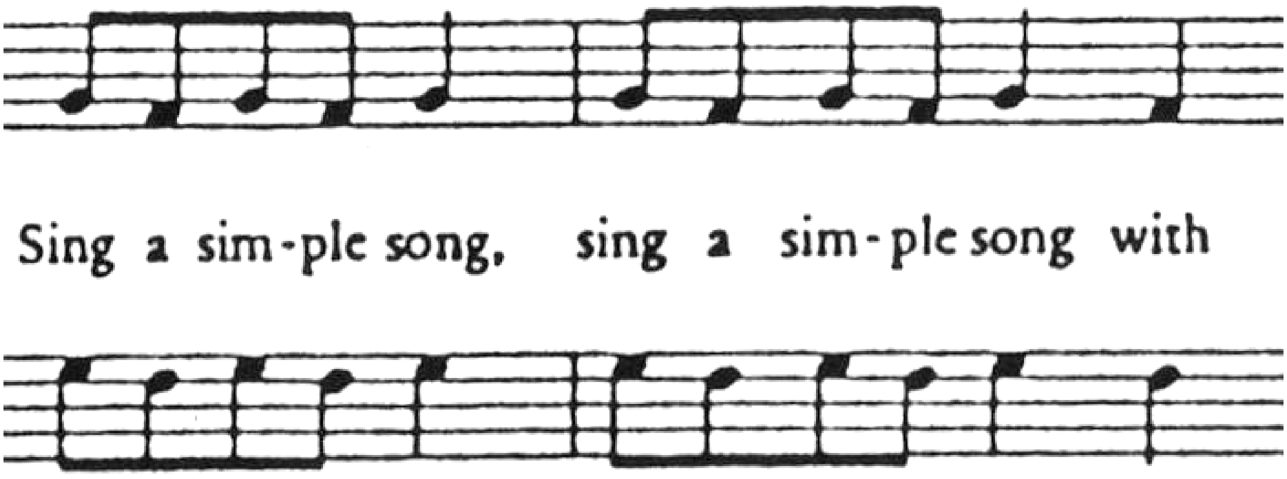
Adjust Brightness and Contrast

The simplest way is to adjust the brightness of an input image. Although Audiveris has a very good automatic binarization algorithm, sometimes manual adjustment improves the recognition.

Have a look a the following part of a score: there are a lot of disturbances between the staff lines.

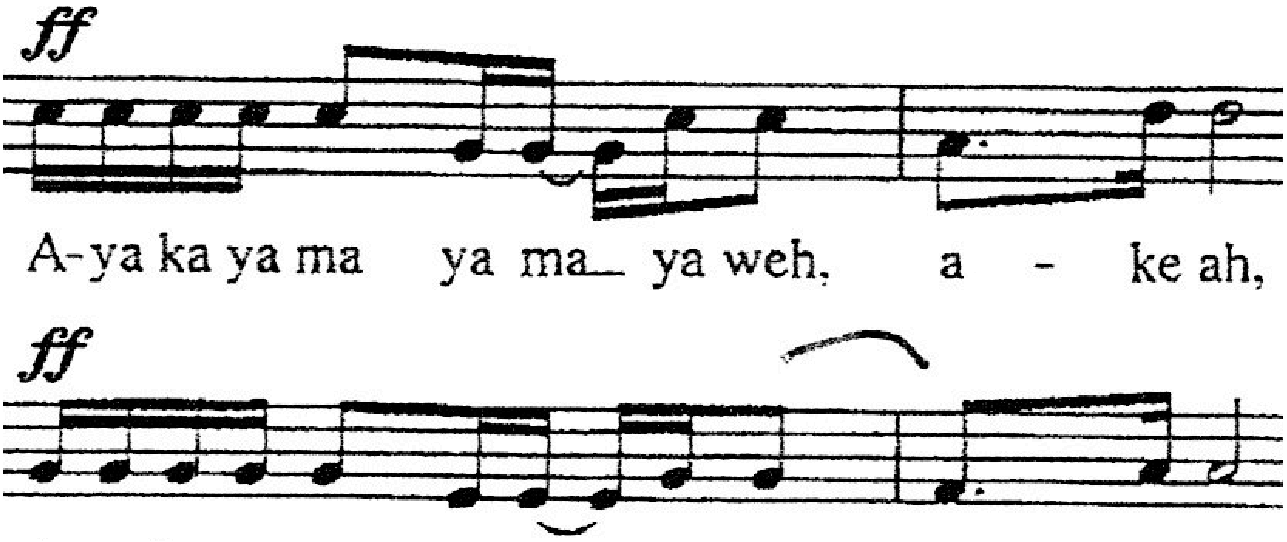


Now simply increase the brightness of the image by using the color curve tool. Keep the dark parts black, to get an image like this:

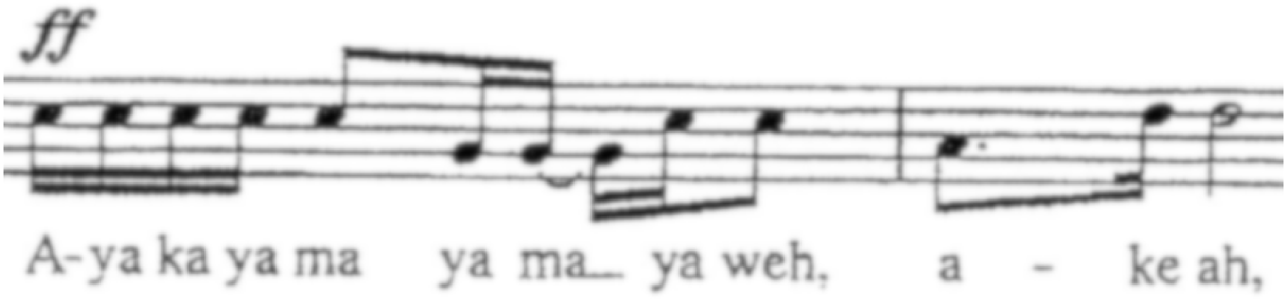


Improve Image using Filters

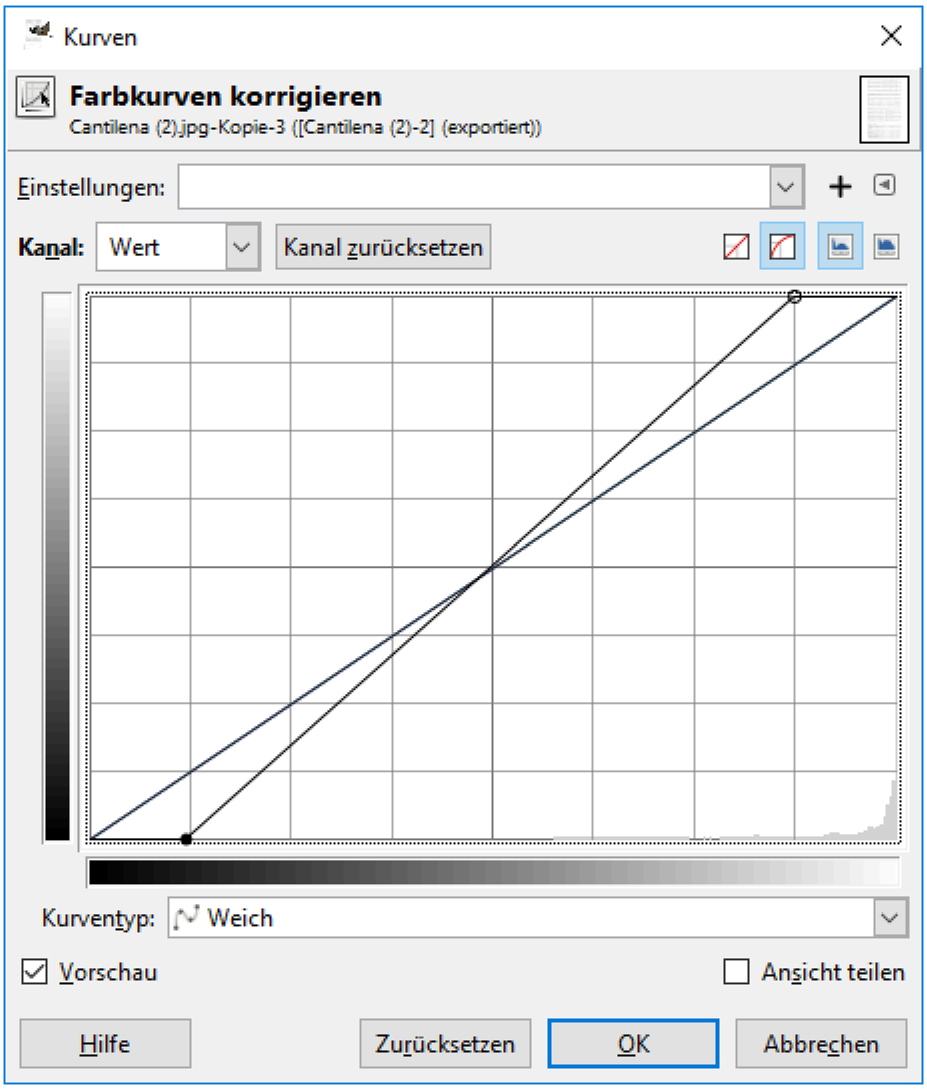
Have a look at the following image: here we have a lot of noise in the lines and in the bars. The transcriptions will have problem to properly detect the bars in such a case.



Now use a gaussian blur filter with size 1.5 to 2.0, and you get the following:



Now use the color curve tool an remove about the top and bottom 10% of the input brightness range (look at the result, the optimal values depend on your input image):



Kurven

Farbkurven korrigieren

Cantilena (2).jpg-Kopie-3 ([Cantilena (2)-2] (exportiert))

Einstellungen:

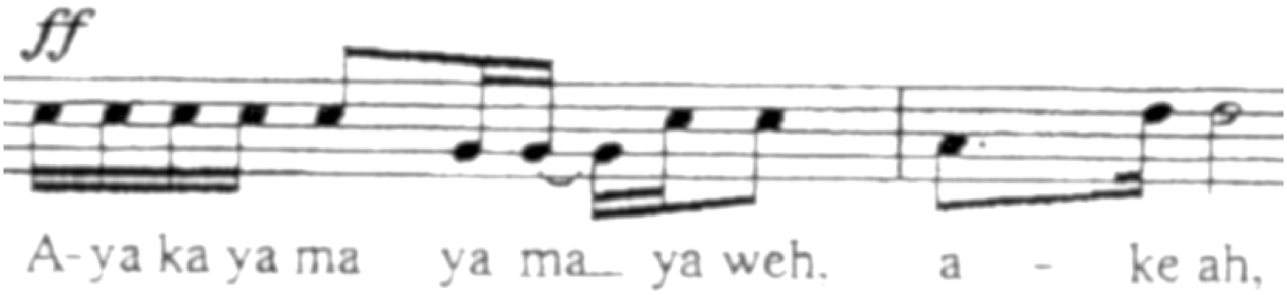
Kanal: Wert

Kurventyp: Weich

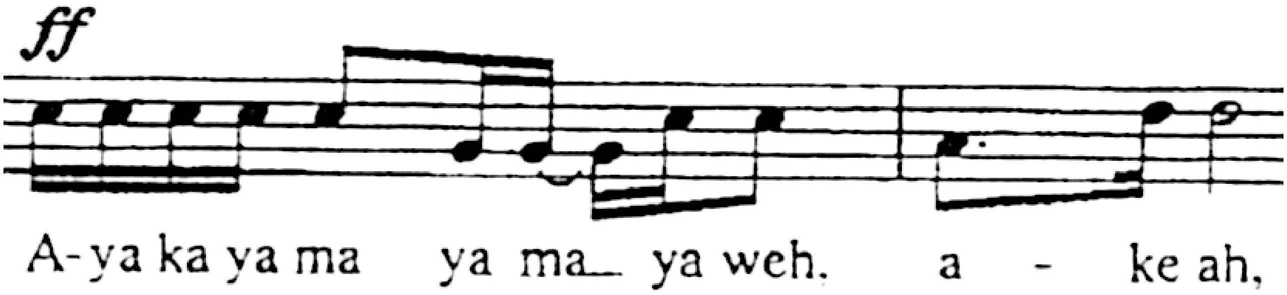
☑ Vorschau

☐ Ansicht teilen

and you get something like this (the stems should still be a close line):



Finally use an “unsharp mask” filter with standard deviation set to about 1.0 to regenerate a sharp image:



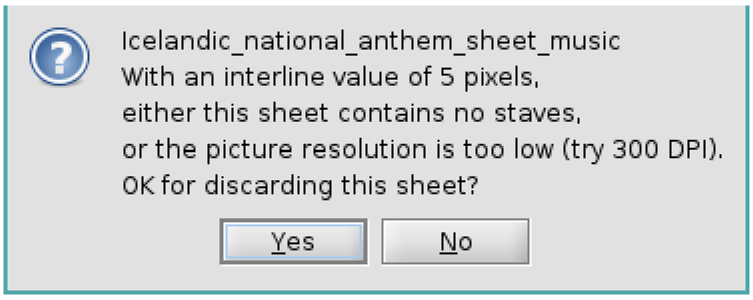
You see that now the noise is almost completely removed.

Enlarging Low Resolution Scans

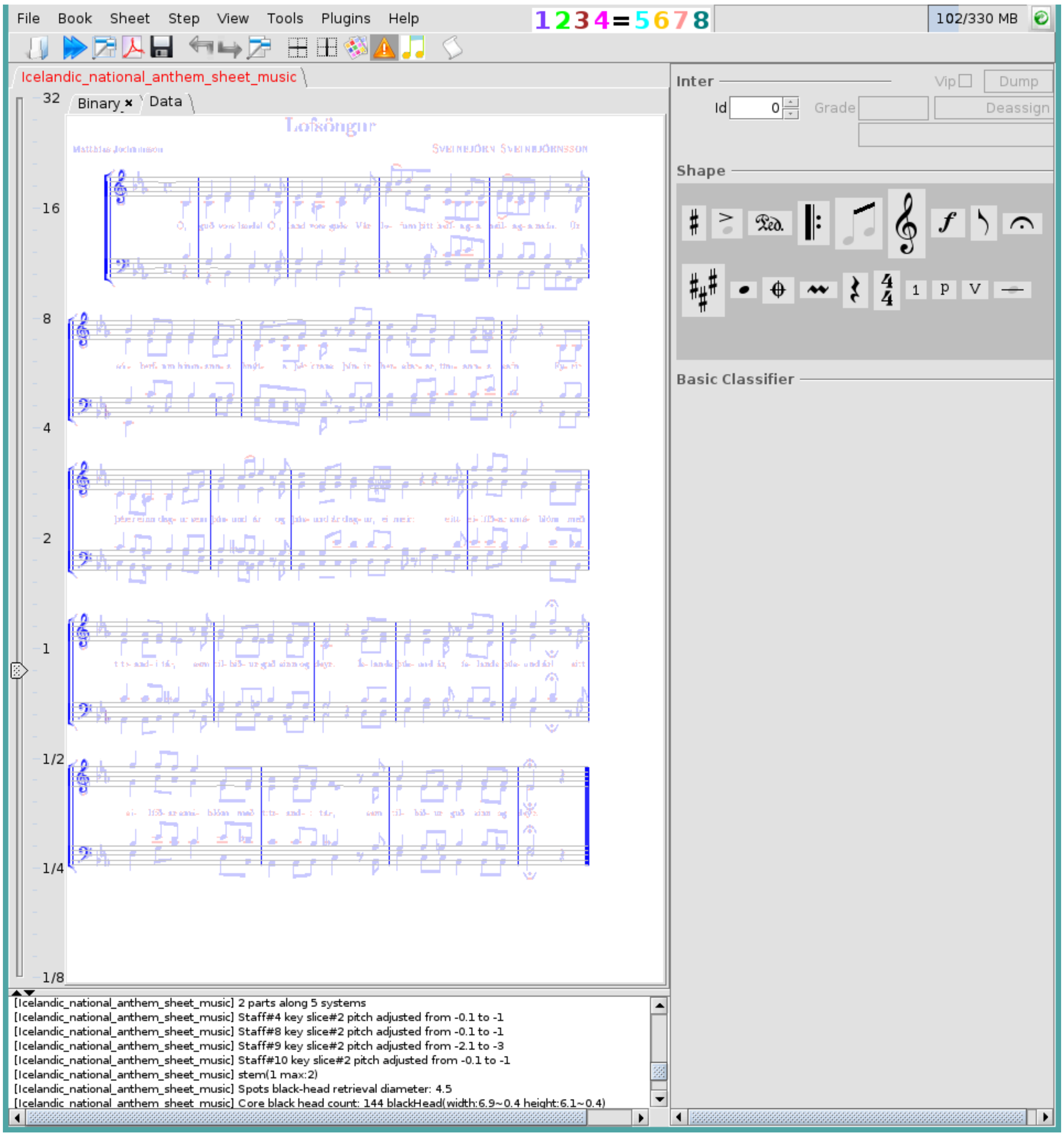
[Contribution by Ram Kromberg]

This section describes how to pre-process low resolution sheet music scans before processing with Audiveris OMR.

When attempting to process low-resolution images with Audiveris, one may encounter warnings and/or identification errors.

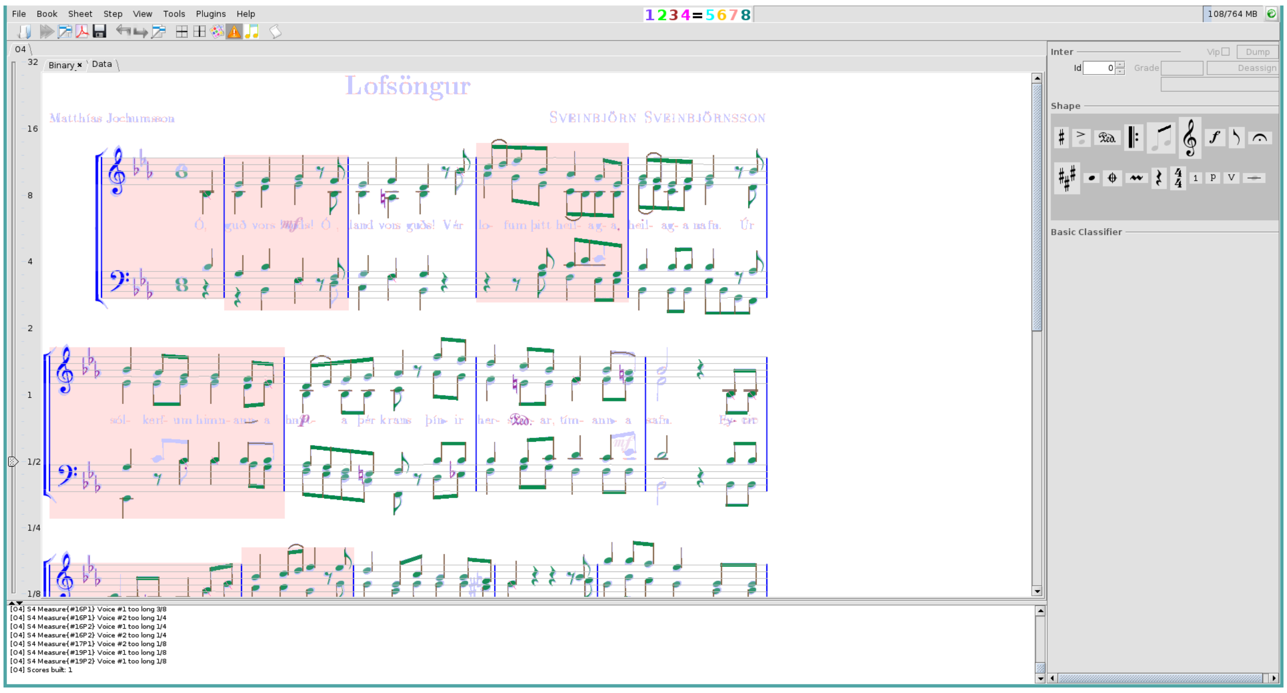


Icelandic_national_anthem_sheet_music
With an interline value of 5 pixels,
either this sheet contains no staves,
or the picture resolution is too low (try 300 DPI).
OK for discarding this sheet?



```
[Icelandic_national_anthem_sheet_music] 2 parts along 5 systems
[Icelandic_national_anthem_sheet_music] Staff#4 key slice#2 pitch adjusted from -0.1 to -1
[Icelandic_national_anthem_sheet_music] Staff#8 key slice#2 pitch adjusted from -0.1 to -1
[Icelandic_national_anthem_sheet_music] Staff#9 key slice#2 pitch adjusted from -2.1 to -3
[Icelandic_national_anthem_sheet_music] Staff#10 key slice#2 pitch adjusted from -0.1 to -1
[Icelandic_national_anthem_sheet_music] stem(1 max:2)
[Icelandic_national_anthem_sheet_music] Spots black-head retrieval diameter: 4.5
[Icelandic_national_anthem_sheet_music] Core black head count: 144 blackHead(width:6.9~0.4 height:6.1~0.4)
```

It is possible to improve or even entirely overcome such obstacles using super-resolution processing software and services to enlarge the image before processing with Audiveris. e.g.



```
[04] S4 Measure{#16P1} Voice #1 too long 3/8
[04] S4 Measure{#16P1} Voice #2 too long 1/4
[04] S4 Measure{#16P2} Voice #1 too long 1/4
[04] S4 Measure{#16P2} Voice #2 too long 1/4
[04] S4 Measure{#17P1} Voice #2 too long 1/8
[04] S4 Measure{#19P1} Voice #1 too long 1/8
[04] S4 Measure{#19P2} Voice #1 too long 1/8
[04] Scores built: 1
```

This document will detail an example using free, open-source software that is commonly distributed in Linux distributions. Alternatives for Macintosh and Windows will be mentioned as well.

Software Required

1 *waifu2x*. For the enlargement.

- Linux: waifu2x-ncnn-vulkan.
- Windows: waifu2x_snowshell.
- Mac: waifu2x-ncnn-vulkan-macOS.

1 ImageMagick. For git-2-png conversion.

2 *wget*. To download the example image.

3 Audiveris OMR.

Step-by-step

1 In this example we'll be using the Icelandic national anthem as follows:

```
wget "https://upload.wikimedia.org/wikipedia/commons/6/6c/Icelandic_national_anthem_sheet_music.
```

2 Since *waifu2x-ncnn-vulkan* can't process gif files, we'll convert the image to a png using ImageMagick's *convert* executable:

```
convert "is~.gif" 00.png
```

3 Finally, we'll enlarge the converted png:

```
waifu2x-ncnn-vulkan -s 4 -i 00.png -o 04.png
```

Notes

1

By default, *waifu2x-ncnn-vulkan* makes use of the GPU. It is possible to force the use of the CPU with the `-g -1` flag and specify the use of more threads with the `-j` flag. e.g. 4 CPU threads: `waifu2x-ncnn-vulkan -g -1 -j 2:4:4 -s 2 -i 00.png -o 04.png`. Be advised even low-end GPUs will out-perform the CPU many times over.

2

In this example we were forced to override the default scaling to 4x using `-s 4` to satisfy Audiveris requirements. There will be times when the default 2x would suffice. Other-times, 4x will prove too much as we'll cross Audiveris's maximum pixel capacity.

3

When executed to scaling exceeding 2x, *waifu2x-ncnn-vulkan* will often terminate with `Segmentation fault (core dumped)` but would otherwise enlarge the images sufficiently well.

4

waifu2x-ncnn-vulkan comes with denoising capabilities that can be used with or without scaling. That is, a severely degraded or poorly focused image may still be recoverable for OMR use as long as it is readable by humans.

Constants

There is (or should be) no hard-coded constant in Audiveris code. Instead, algorithms are backed by “*application constants*” (more than 800 of them today). This mechanism is a low level yet powerful way to handle nearly all application tuning data.

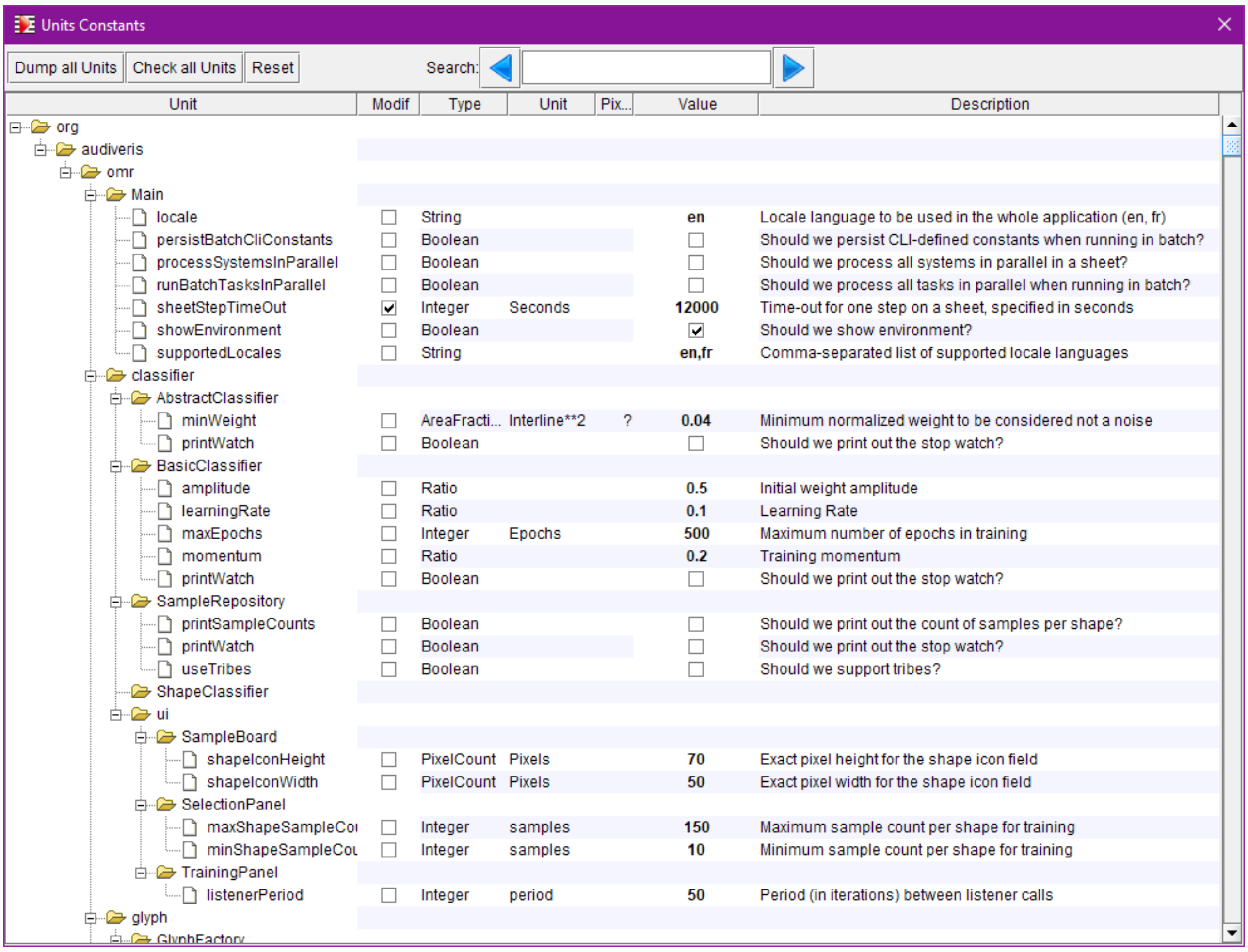
This data is presented as constants to the end user, and is modifiable at run time:

- Through the CLI `-constant KEY=VALUE` argument
- Through the pull-down menu `Tools → Constants`

TABLE OF CONTENTS

Dialog

The display below combines a tree of classes on the left side, and a table on the right side, where details of the constants from the containing classes are available for display and modification.



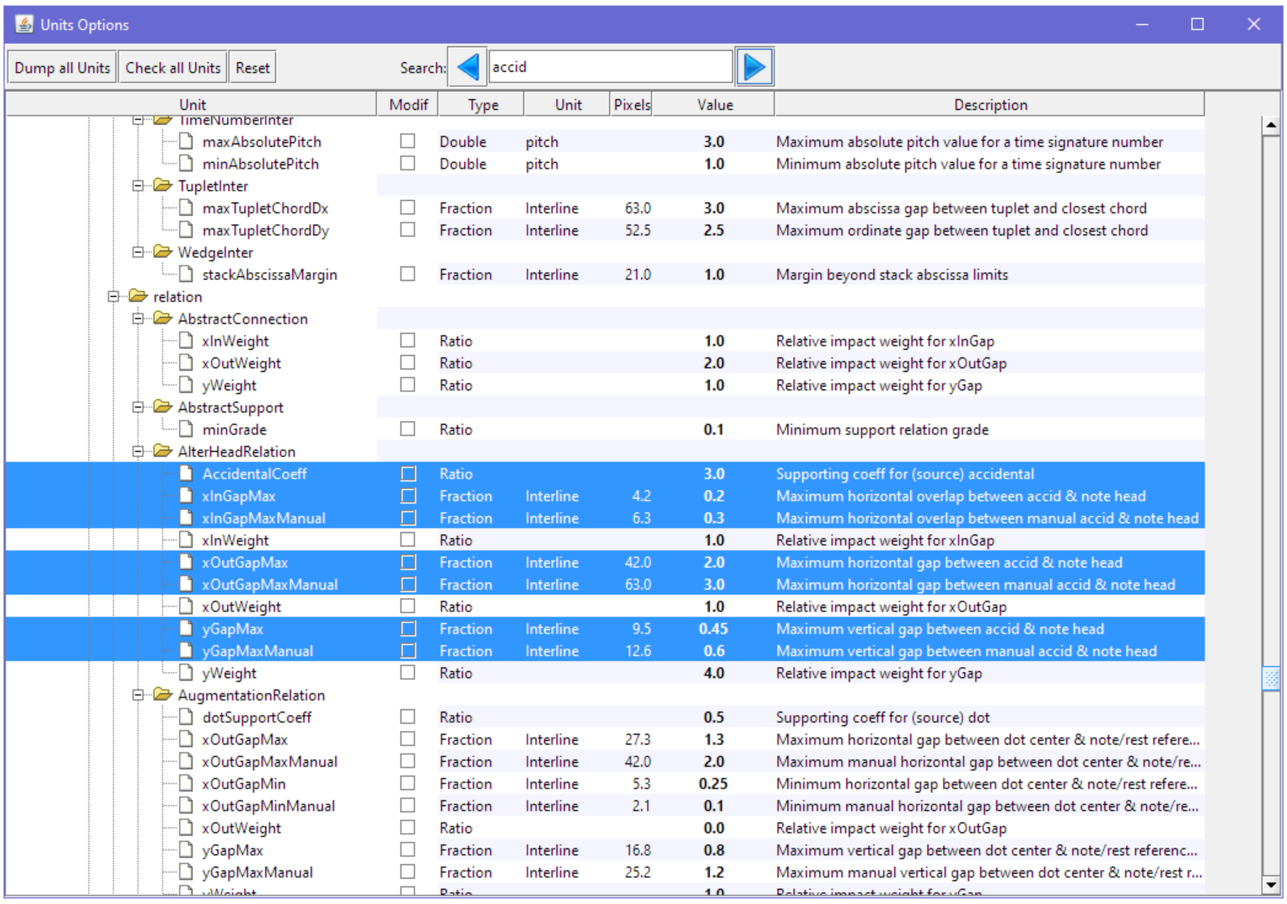
Unit	Modif	Type	Unit	Pix...	Value	Description
org						
audiveris						
omr						
Main						
locale	☐	String			en	Locale language to be used in the whole application (en, fr)
persistBatchCliConstants	☐	Boolean			☐	Should we persist CLI-defined constants when running in batch?
processSystemsInParallel	☐	Boolean			☐	Should we process all systems in parallel in a sheet?
runBatchTasksInParallel	☐	Boolean			☐	Should we process all tasks in parallel when running in batch?
sheetStepTimeOut	☑	Integer	Seconds		12000	Time-out for one step on a sheet, specified in seconds
showEnvironment	☐	Boolean			☑	Should we show environment?
supportedLocales	☐	String			en,fr	Comma-separated list of supported locale languages
classifier						
AbstractClassifier						
minWeight	☐	AreaFracti...	Interline**2	?	0.04	Minimum normalized weight to be considered not a noise
printWatch	☐	Boolean			☐	Should we print out the stop watch?
BasicClassifier						
amplitude	☐	Ratio			0.5	Initial weight amplitude
learningRate	☐	Ratio			0.1	Learning Rate
maxEpochs	☐	Integer	Epochs		500	Maximum number of epochs in training
momentum	☐	Ratio			0.2	Training momentum
printWatch	☐	Boolean			☐	Should we print out the stop watch?
SampleRepository						
printSampleCounts	☐	Boolean			☐	Should we print out the count of samples per shape?
printWatch	☐	Boolean			☐	Should we print out the stop watch?
useTribes	☐	Boolean			☐	Should we support tribes?
ShapeClassifier						
ui						
SampleBoard						
shapeIconHeight	☐	PixelCount	Pixels		70	Exact pixel height for the shape icon field
shapeIconWidth	☐	PixelCount	Pixels		50	Exact pixel width for the shape icon field
SelectionPanel						
maxShapeSampleCo	☐	Integer	samples		150	Maximum sample count per shape for training
minShapeSampleCou	☐	Integer	samples		10	Minimum sample count per shape for training
TrainingPanel						
listenerPeriod	☐	Integer	period		50	Period (in iterations) between listener calls
glyph						
GlyphFactory						

The picture represents the top of the scrollable Constants view: We are located at the top of Audiveris software, with the root packages: `org`, `audiveris`, `omr`.

To ease the browsing of all constants, there is a `Search` area, where the user can enter some text to be searched for among the class names, constant names and constant descriptions. The search is not case-sensitive.

Let’s suppose we would like to somewhat relax the constraint on the horizontal distance between an accidental alteration sign (like a sharp) and the note head on its right side.

We can enter `accid` in the search area and press the search button (perhaps several times) until we get to several interesting constants. Here is what we can read:



Unit	Modif	Type	Unit	Pixels	Value	Description
TimeNumberInter						
maxAbsolutePitch	☐	Double	pitch		3.0	Maximum absolute pitch value for a time signature number
minAbsolutePitch	☐	Double	pitch		1.0	Minimum absolute pitch value for a time signature number
TupletInter						
maxTupletChordDx	☐	Fraction	Interline	63.0	3.0	Maximum abscissa gap between tuplet and closest chord
maxTupletChordDy	☐	Fraction	Interline	52.5	2.5	Maximum ordinate gap between tuplet and closest chord
WedgeInter						
stackAbscissaMargin	☐	Fraction	Interline	21.0	1.0	Margin beyond stack abscissa limits
relation						
AbstractConnection						
xInWeight	☐	Ratio			1.0	Relative impact weight for xInGap
xOutWeight	☐	Ratio			2.0	Relative impact weight for xOutGap
yWeight	☐	Ratio			1.0	Relative impact weight for yGap
AbstractSupport						
minGrade	☐	Ratio			0.1	Minimum support relation grade
AlterHeadRelation						
AccidentalCoeff	☐	Ratio			3.0	Supporting coeff for (source) accidental
xInGapMax	☐	Fraction	Interline	4.2	0.2	Maximum horizontal overlap between accid & note head
xInGapMaxManual	☐	Fraction	Interline	6.3	0.3	Maximum horizontal overlap between manual accid & note head
xInWeight	☐	Ratio			1.0	Relative impact weight for xInGap
xOutGapMax	☐	Fraction	Interline	42.0	2.0	Maximum horizontal gap between accid & note head
xOutGapMaxManual	☐	Fraction	Interline	63.0	3.0	Maximum horizontal gap between manual accid & note head
xOutWeight	☐	Ratio			1.0	Relative impact weight for xOutGap
yGapMax	☐	Fraction	Interline	9.5	0.45	Maximum vertical gap between accid & note head
yGapMaxManual	☐	Fraction	Interline	12.6	0.6	Maximum vertical gap between manual accid & note head
yWeight	☐	Ratio			4.0	Relative impact weight for yGap
AugmentationRelation						
dotSupportCoeff	☐	Ratio			0.5	Supporting coeff for (source) dot
xOutGapMax	☐	Fraction	Interline	27.3	1.3	Maximum horizontal gap between dot center & note/rest refere...
xOutGapMaxManual	☐	Fraction	Interline	42.0	2.0	Maximum manual horizontal gap between dot center & note/re...
xOutGapMin	☐	Fraction	Interline	5.3	0.25	Minimum horizontal gap between dot center & note/rest refere...
xOutGapMinManual	☐	Fraction	Interline	2.1	0.1	Minimum manual horizontal gap between dot center & note/re...
xOutWeight	☐	Ratio			0.0	Relative impact weight for xOutGap
yGapMax	☐	Fraction	Interline	16.8	0.8	Maximum vertical gap between dot center & note/rest referenc...
yGapMaxManual	☐	Fraction	Interline	25.2	1.2	Maximum manual vertical gap between dot center & note/rest r...
yWeight	☐	Ratio			1.0	Relative impact weight for yGap

1. We are in the package named `relation`, (its full name is org.audiveris.omr.sig.relation) and the class named `AlterHeadRelation`, which governs a potential relation between an alteration and a head.
2. Among the various class constants, we are interested by `xOutGapMax` and `xOutGapMaxManual` which define the maximum horizontal gap between an accidental alteration and a note head. The former is meant for the OMR engine, the latter for manual assignment. Threshold for manual assignment is larger, because we can offer to be less strict when the user is in the driver’s seat.
3. Notice that these value fields are coded using “interline” fractions, (2.0 and 3.0 on this picture) to be scale independent. If we have a selected sheet at hand, the previous column (`Pixels`) displays the corresponding number of pixels using the sheet interline scale (42 and 63 on this picture).
4. We can directly select the value field by a double-click and then type a new value there. The new value applies immediately.

Constants Lifecycle

The overriding mechanism is defined as follows, from lower to higher priority:

1. FACTORY: The default value as defined in the source code,
2. USER: The modified value, if any, persisted in a specific `run.properties` file located in the user config folder,

3. CLI: The value, if any, specified in the command line by a `-constant key=value`. The CLI value is persisted in the USER file when running in interactive mode, and not persisted when running in batch.
4. UI: The value, if any, specified through the `Tools → Constants` user interface. These UI values are persisted in the USER file.

If ever we want to discard the modification we have made and get back to the FACTORY value of an constant, we simply uncheck the box in the `Modif` column of this constant in the Constants window.

To restore the FACTORY value for **all** constants, we use the `Reset` button located at the top of the Constants window. Needless to say, we will be prompted for confirmation beforehand…

Plugins

Many music notation programs, if not all, can import MusicXML files as those exported by Audiveris.

The standard sequence is to:

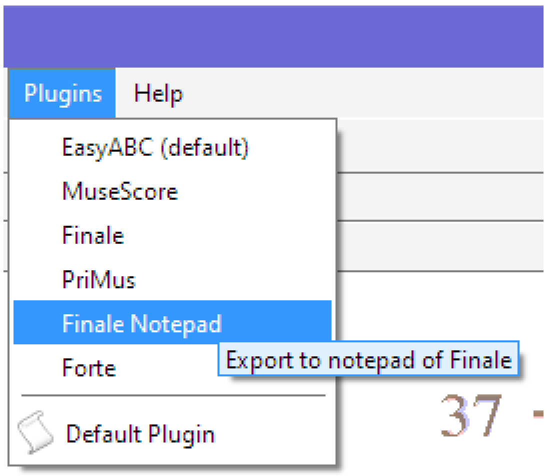
1. Ask Audiveris to transcribe the book at hand, if so needed,
2. Export book music into a MusicXML file,
3. Launch the desired external program,
4. Ask the external program to import MusicXML,
5. Make it navigate to where the MusicXML file is located.

This sequence can be automated via the use of one or several plugins.

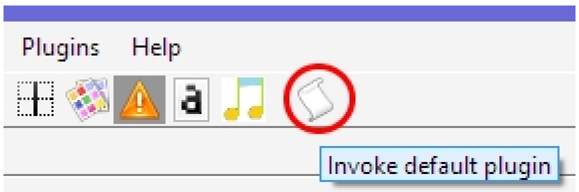
TABLE OF CONTENTS

Use of Plugins

Once plugins are correctly configured, we can call an external program by selecting the desired one in the pull-down `Plugins` menu:



Even simpler, the default plugin can be directly called by pressing the toolbar plugin icon:



In either case, Audiveris makes sure the book has been transcribed, that the MusicXML export file exists and is up-to-date with the latest score modifications performed manually by the user, and finally launches the proper external program, providing the path to the exported file as an import argument.

Plugins Configuration

We can define one or more plugins, by creating a single XML file, named `plugins.xml`, in the Audiveris configuration folder.

At Audiveris launch time, if a `plugins.xml` file is found in the configuration folder, then a proper pull-down `Plugins` menu and a plugin toolbar button are defined according to the file content.

Here below is an example of such `plugins.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
===================================================================================================
                                        p l u g i n s . x m l
===================================================================================================
    This file will be looked up in user config folder.
    It defines the list of Audiveris export plugins.
    Each plugin element is defined as follows:
    - Attribute id: (mandatory) Unique name for the plugin
    - Attribute tip: (optional) Description to be used as a tip
    - Elements arg: (mandatory) One separate element for each argument.
                    At least one arg element must equal {} to indicate where to insert at run-time
                    the export file path(s).
===================================================================================================
-->
<plugins>

    <!-- MuseScore -->
    <plugin id="MuseScore" tip="Invoke MuseScore on score XML">
        <arg>C:\Program Files\MuseScore 3\bin\MuseScore3.exe</arg>
        <arg>{}</arg>
    </plugin>

    <!-- Finale -->
    <plugin id="Finale" tip="Export to Finale">
        <arg>C:\Program Files (x86)\Finale 2014\Finale.exe</arg>
        <arg>{}</arg>
    </plugin>

    <!-- Finale Notepad -->
    <plugin id="Finale Notepad" tip="Export to notepad of Finale">
        <arg>C:\Program Files (x86)\Finale NotePad 2012\Finale NotePad.exe</arg>
        <arg>{}</arg>
    </plugin>

    <!-- PriMus -->
    <plugin id="PriMus" tip="Use of PriMus">
        <arg>C:\Program Files (x86)\PriMus Demo\PriMus.exe</arg>
        <arg>{}</arg>
    </plugin>

    <!-- EasyABC -->
    <plugin id="EasyABC" tip="Export to EasyABC">
        <arg>C:\Program Files (x86)\EasyABC\easy_abc.exe</arg>
        <arg>{}</arg>
    </plugin>

    <!-- Forte -->
    <plugin id="Forte" tip="Forte 9 Free">
        <arg>C:\Program Files (x86)\FORTE\FORTE 9 Free\FORTE.exe</arg>
        <arg>{}</arg>
    </plugin>

</plugins>
```

Although this `plugins.xml` example is obviously meant for a Windows environment, its content is rather self-explanatory:

- The first line is the mandatory signature for an XML file.
- Then a `plugins` element will contain one or several `plugin` children.
- Each `<plugin>` child defines a separate plugin.

- The first plugin found in file is taken as the `default` one.
- The sequence of `<arg>` elements defines the precise syntax of a CLI command.
- The `{}` arg is a placeholder where Audiveris will insert the path(s) to the exported MusicXML file(s).

Samples

This chapter describes how to set up repositories of Glyph samples and how to carefully filter the global repository so that it can be used for training.

Note: To use this feature, make sure the advanced `SAMPLES` topic is activated. If not, select the `SAMPLES` topic in the menu (and restart the application).

Audiveris basic classifier is a Glyph classifier, which means that it needs to be trained with glyph samples (a sample is basically a glyph with the related shape).

TABLE OF CONTENTS

Initial samples

There is an Audiveris Google drive located at this address.

It contains training data, organized by folders. Let’s inspect folder 5.3 which is the more recent folder as of this writing.

This folder contains three files:

File Name	Size	Content
`samples.zip`	4 MB	~15000 samples
`images.zip`	453 MB	~3500 sheet

File Name	Size	Content
		images
`basic-classifier.zip`	0.3 MB	Resulting trained model

The `samples.zip` file is the `global` repository. It’s the one that has been used to train the Glyph classifier provided with latest Audiveris release.

We begin by downloading this archive (without expanding it) into our own `config/train` folder (see its precise location in the Essential Folders section). We will then be able to augment this collection on our own.

We can also download the `images.zip` file which is not mandatory for training, but which will help us see most samples within their sheet context.

Sampling a Sheet or a Book

After perhaps some manual corrections, when we are really satisfied with all the glyphs recognized in a given sheet, we can save the sheet data as training samples. This is done via the pull-down menu `Sheet → Sample sheet symbols`.

We can also use the pull-down menu `Book → Sample book symbols` if we are comfortable with all the book sheets.

Data is saved under two zip archives located in the book folder. In say the `foo` book folder, there will be:

- `foo-samples.zip` for the collection of samples (this is the book sample repository)
- `foo-images.zip` for the containing sheet images.

It is a good practice to work on one book at a time, and only merge a book repository into the global repository when the book data has been thoroughly verified and its samples carefully filtered.

Sample Repository

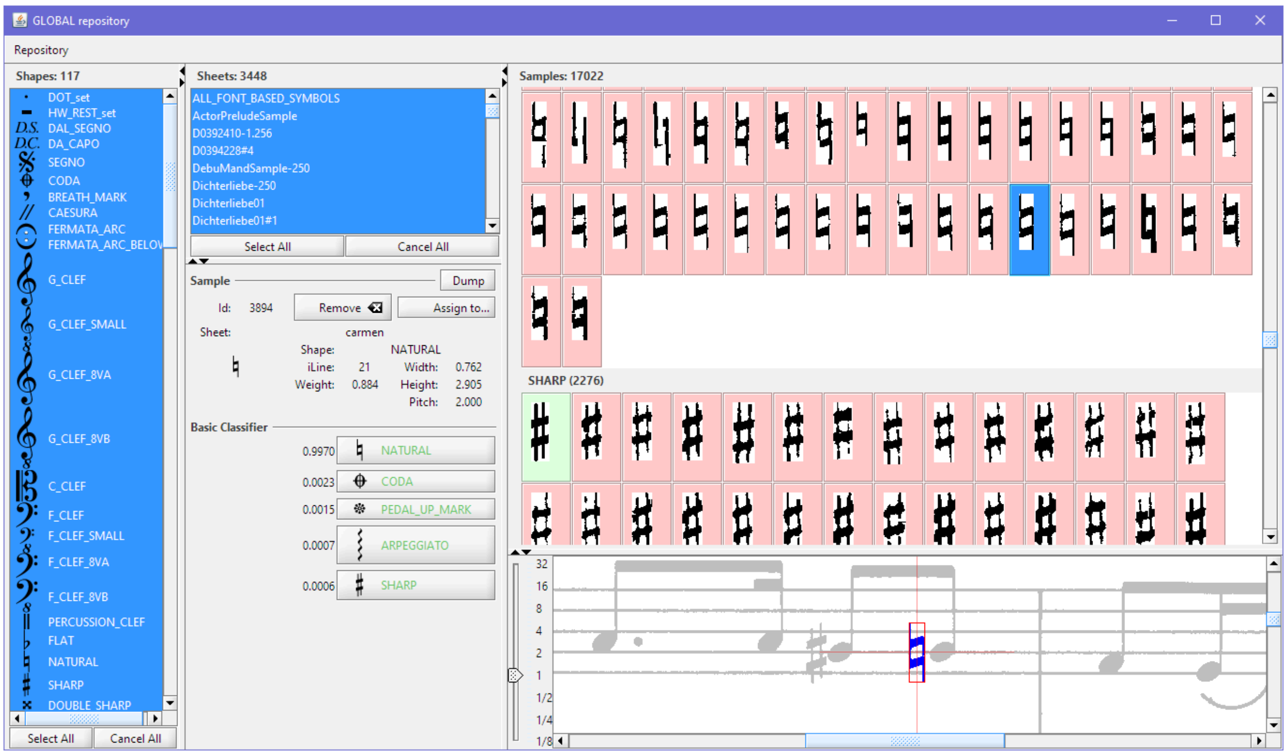
The purpose of the Sample Repository dialog is to provide a user interface to visually review some or all of the various SAMPLES which could be used for training the classifier.

The main objective is to easily identify samples which have been assigned a wrong shape. Using them as part of the training base could severely impact Audiveris recognition efficiency. So the strategy, when such a wrong sample has been identified, is to remove it from the repository or to assign it a correct shape.

To work on the **GLOBAL** repository, we use the `Tools → Browse Global Repository...` pull-down menu. It will work on the `samples.zip` and `images.zip` files if any are found in our own `config/train` folder (see the Train folder section).

To work on a **book** repository, we use the `Book → View Book Repository...` pull-down menu. We will be able to pick and merge book repositories into the GLOBAL repository later on, because the training process uses only the GLOBAL repository.

Either way, the repository user interface is the same:



Id: 3894

Sheet: carmen

Shape: NATURAL

iLine: 21 Width: 0.762

Weight: 0.884 Height: 2.905

Pitch: 2.000

0.9970 NATURAL

0.0023 CODA

0.0015 PEDAL_UP_MARK

0.0007 ARPEGGIATO

0.0006 SHARP

The repository interface is organized as follows:

- Left column:
  - Shapes selector
- Middle column:
  - Sheets selector
  - Sample board
  - Classifier board
- Right column:
  - Samples selector
  - Sample context

Sample Selection

1. Initially, all panes are empty except the `Sheets` selector which appears populated with sheets names.
   These are the sheets available in the underlying repository (either the book sheets for a book repository, or all registered sheets so far for the global repository).
   There are a few special sheets, all prefixed by `# SYMBOLS FROM FONT #`, which are not real sheets, but gather all the synthetic samples built from Audiveris musical and textual font families.
   We can select one or several sheets in the Sheets selector.

2. The `Shapes` selector gets populated as soon as sheets are selected.
   We can now select the shapes of interest.

3. The `Samples` selector gets populated as soon as shapes get selected.
   The samples are gathered by shape. In a shape collection, via a right-click, we can sort samples by width, height, height or grade.
   The synthetic samples (there are a handful of these in each shape collection) are displayed with a green background.
   Only one sample can be selected at a time.

4. If an image is available for the sheet which contains the sample at hand, the Sample `context` panel displays the selected sample in its sheet context.
   This can be helpful for visual checking.

Sample Editing

The selected sample can be:

- **Removed** from the repository.
  This is done by typing the `DELETE` key or clicking on the `Remove` button in the Sample board or selecting the `Remove` item in the sample right-click menu.
- **Assigned** to a new shape.
  This is done by clicking on the `Assign to` button in the Sample board or selecting the `Assign to` item in the sample right-click menu, and then selecting the new shape.

Merging Repositories

When we are satisfied with a book repository we can push its content to the global repository.

We do so from the book repository interface, by selecting the pull-down menu `Repository → Push to Global`.

Training

Audiveris has the ability to train the underlying Glyph classifier with representative samples.

Note that the program is released with a pre-trained classifier so the casual user can safely ignore this training section.

However, if the score(s) we want to transcribe use some specific music font significantly different from the provided examples, we may consider training the classifier to better fit our case.

Let’s make sure we have enabled the `SAMPLES` topic in the menu, and restarted the application to take this advanced topic into account in all UI corners.

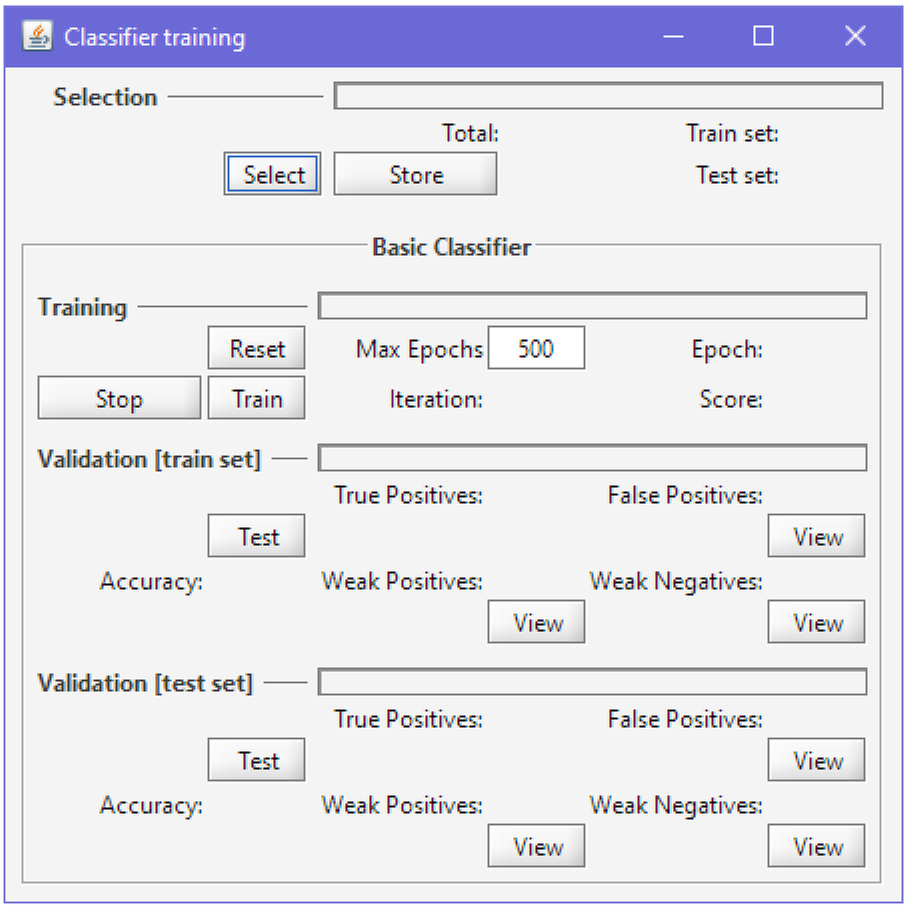
Then, we will need a bunch of training samples (a sample is basically a glyph and a shape). This is addressed in the Samples section before.

Finally, we can launch one or several trainings of the glyph classifier, via the dedicated Trainer dialog.

TABLE OF CONTENTS

Trainer Dialog

This dialog is dedicated to the training of Audiveris basic classifier (a glyph classifier). It is launched via the pull-down menu `Tools → Train classifier` or, from the global repository, by its `Repository → Train classifier` menu.



Max Epochs 500

Here we can launch and monitor the training of the classifier neural network.

The `Select` button makes a new random selection among the samples of the global repository.

The `Reset` button builds a new network from scratch (forgetting any previous training).

The `Train` button launches a training of the current network (which is either the initially-trained one, or a brand new one if `Reset` was hit).

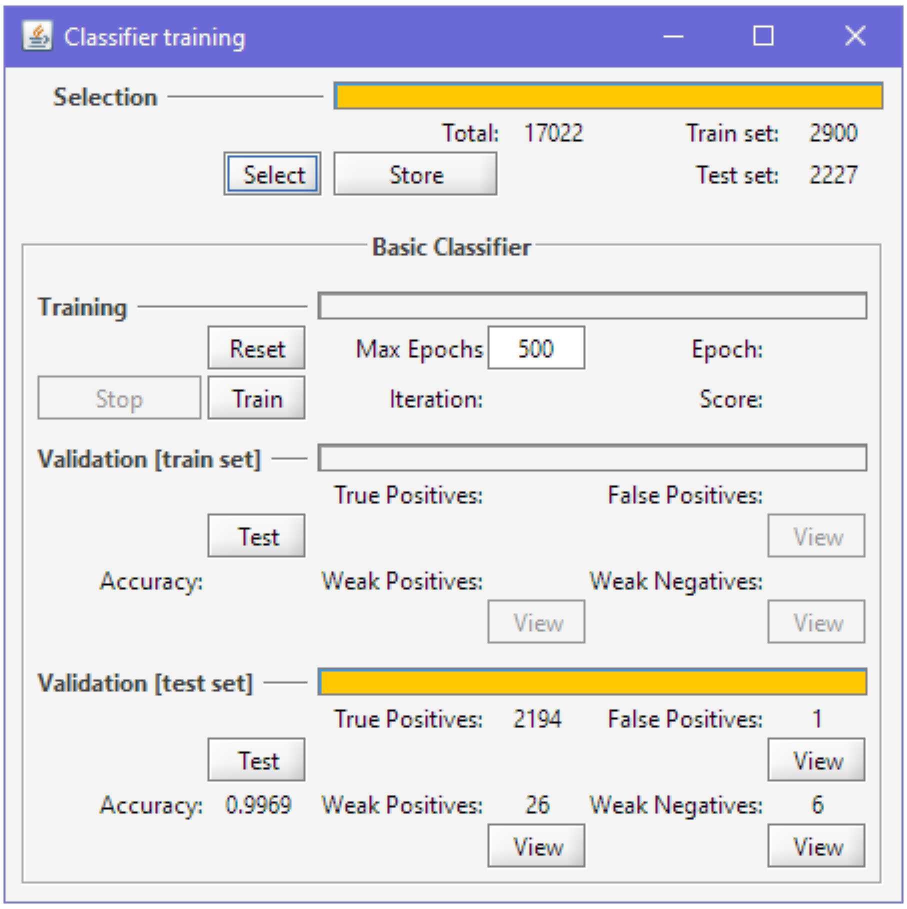
Note that training hyper-parameters cannot be directly modified on this interface. If really needed, we can use the `Tools → Constants` menu and search for `BasicClassifier`. There we can set amplitude, learningRate, momentum and maxEpochs parameters.

The `Stop` button allows to stop the training before the maxEpochs count has been reached.

Validation

There are two data sets available for validation, the train set (of no major interest) and the test set.

The `Test` button launches validation on the data set at hand. Note that validation is disabled if any training is going on.



Selection

Total: 17022 Train set: 2900

Test set: 2227

Basic Classifier

Training

Max Epochs 500 Epoch:

Iteration: Score:

Validation [train set]

True Positives: False Positives:

Accuracy: Weak Positives: Weak Negatives:

Validation [test set]

True Positives: 2194 False Positives: 1

Accuracy: 0.9969 Weak Positives: 26 Weak Negatives: 6

In the picture above, without further training, we have directly run validation on a random test set.

Beside the accuracy measure value, 3 `View` buttons are of interest to launch a repository view with just the samples involved in:

- **False Positives**: samples which were mistaken for a different shape, with a high grade.
- **Weak Positives**, samples correctly recognized, but with a low grade.
- **Weak Negatives**, samples mistaken with a low grade.

Clicking of the `View` button, especially for the False Positive case, allows to manually check if the sample was correct in first place, and if not, remove or reassign it before a potential future (re-)training.

Reference

This `Reference` chapter presents a few key topics that are not mandatory for a first reading.

Each topic is meant to be exhaustive and often provides pointers back to more specific and detailed sections in the handbook presentation.

TABLE OF CONTENTS

Editors

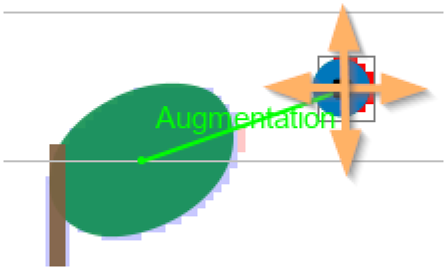
This reference chapter gathers the description of all *`Inter`* editors variants, especially regarding the role and potential move of its handles.

It also presents the two variants (Lines and Global) of the *Staff* editor.

TABLE OF CONTENTS (IN ALPHABETICAL ORDER)

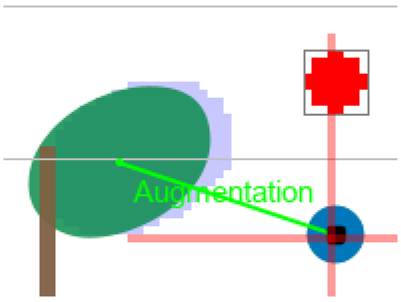
Default `Inter` editor

The default editor applies to every `Inter` class, for example the augmentation dot class here below, unless a more specific editor is defined for the `Inter` class at hand.

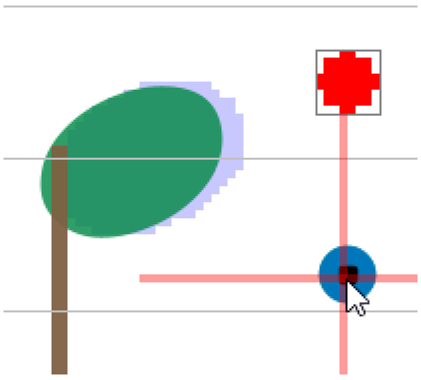


The default editor provides only one handle located at `Inter` center. This very basic editor allows to move the `Inter` in any direction, but provides no way to resize it.

The involved relations if any – like the “*Augmentation*” relation between the note head and the augmentation dot in this example – are dynamically updated while the `Inter` is being moved.



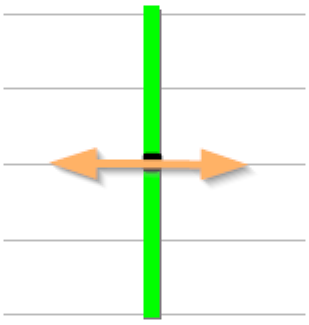
Here, we have moved the augmentation dot one line below and the relation is kept with the augmented note.



Here, we are currently located too far from the note and the relation disappears.

The next sections describe all specific editors, listed in alphabetical order.

Barline/Bracket editor

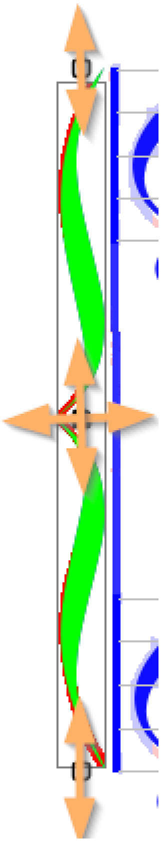


A Barline can move only horizontally, and can’t be resized, because it is **snapped** to staff height.

(In Audiveris data model, a Barline goes from staff top line to staff bottom line. Aligned barlines from different staves are often connected by so-called “Connectors”).

The same editor applies to Bracket as well.

Brace editor

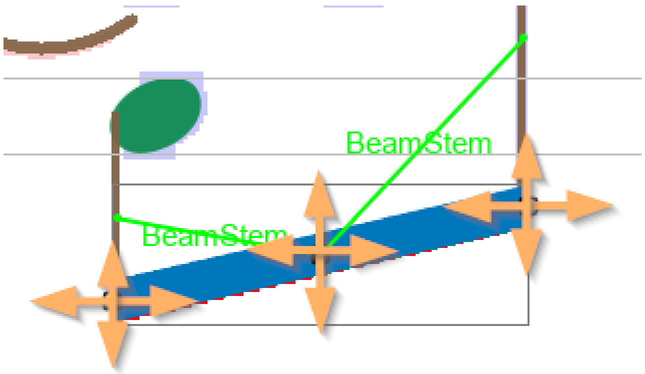


- Center handle shifts the brace in any direction,
- Top and bottom handles resize the brace vertically.

WARNING

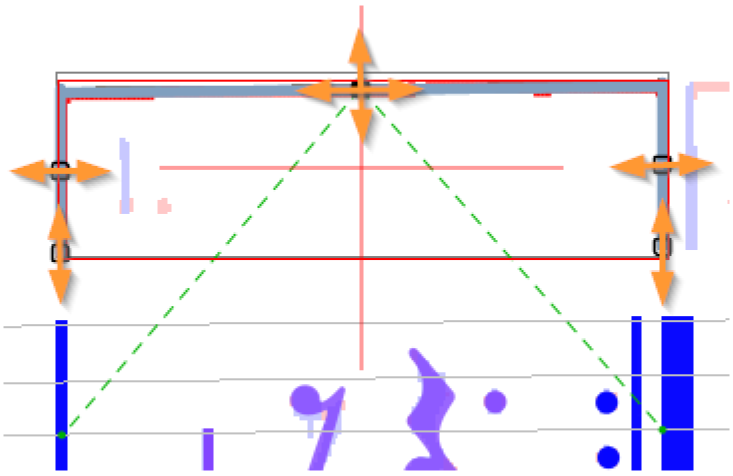
Shifting or resizing a brace is meant for small adjustments only.
If you want to extend or reduce the number of staves embraced by a Brace instance (which is a rather heavy operation that impacts the definition of parts within a system), this must be done **explicitly** by adding or removing a manual Brace. Please refer to Part merge section for such Brace usage.

Beam editor



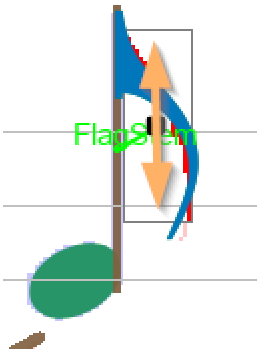
- Center handle moves the whole beam in any direction.
- A side handle moves the side in any direction, **snapping** the beam side on any stem nearby.

Ending editor UPDATED IN 5.8



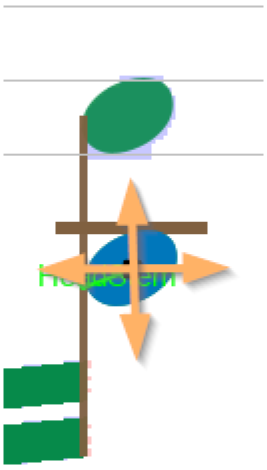
- The center handle moves the whole ending in any direction
- A side handle moves the ending side horizontally (together with its side leg if any)
- A bottom handle resizes the leg vertically

Flag editor



The flag editor is another variation of the default editor with its single center handle, which here can move **only vertically** along the related stem.

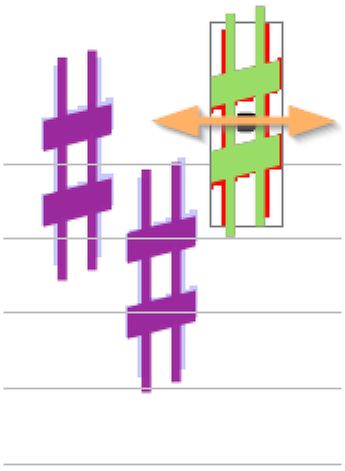
Head editor



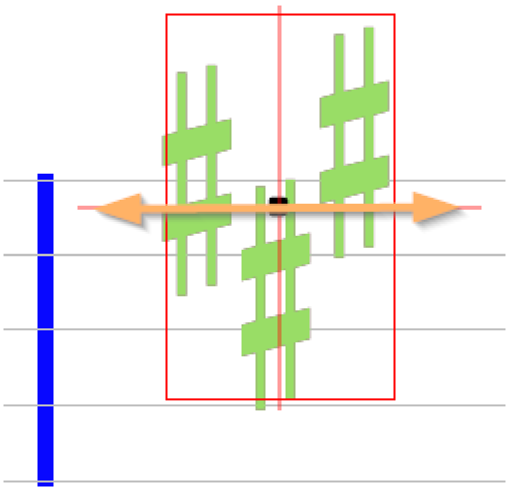
The note head editor is similar to the default editor, with its single center handle. The only difference is that the head being moved is snapped:

- **Vertically** to the underlying staff lines, or ledger lines (perhaps dynamically created while head is shifted away from staff),
- **Horizontally** to the stem nearby, if any, on left or right. Of course, this does not apply to WHOLE or BREVE shapes since these heads use no stem.

KeyAlter/Key editor

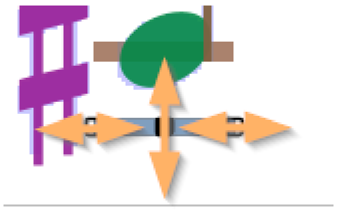


The key alter editor allows to move **horizontally** one KeyAlter member of the key signature.

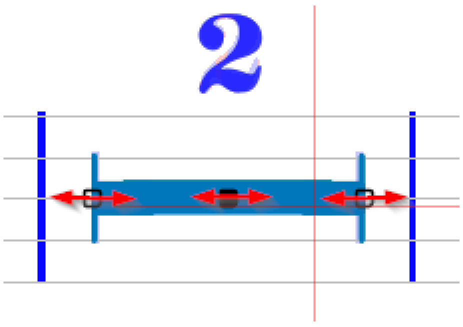


If the key signature is a whole manual signature (e.g. it has been dropped from the ShapePalette), then the editor can shift the **whole key** horizontally.

Ledger editor



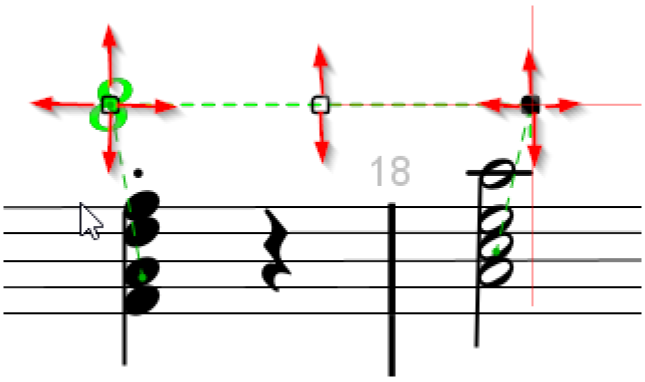
Multi-measure rest editor



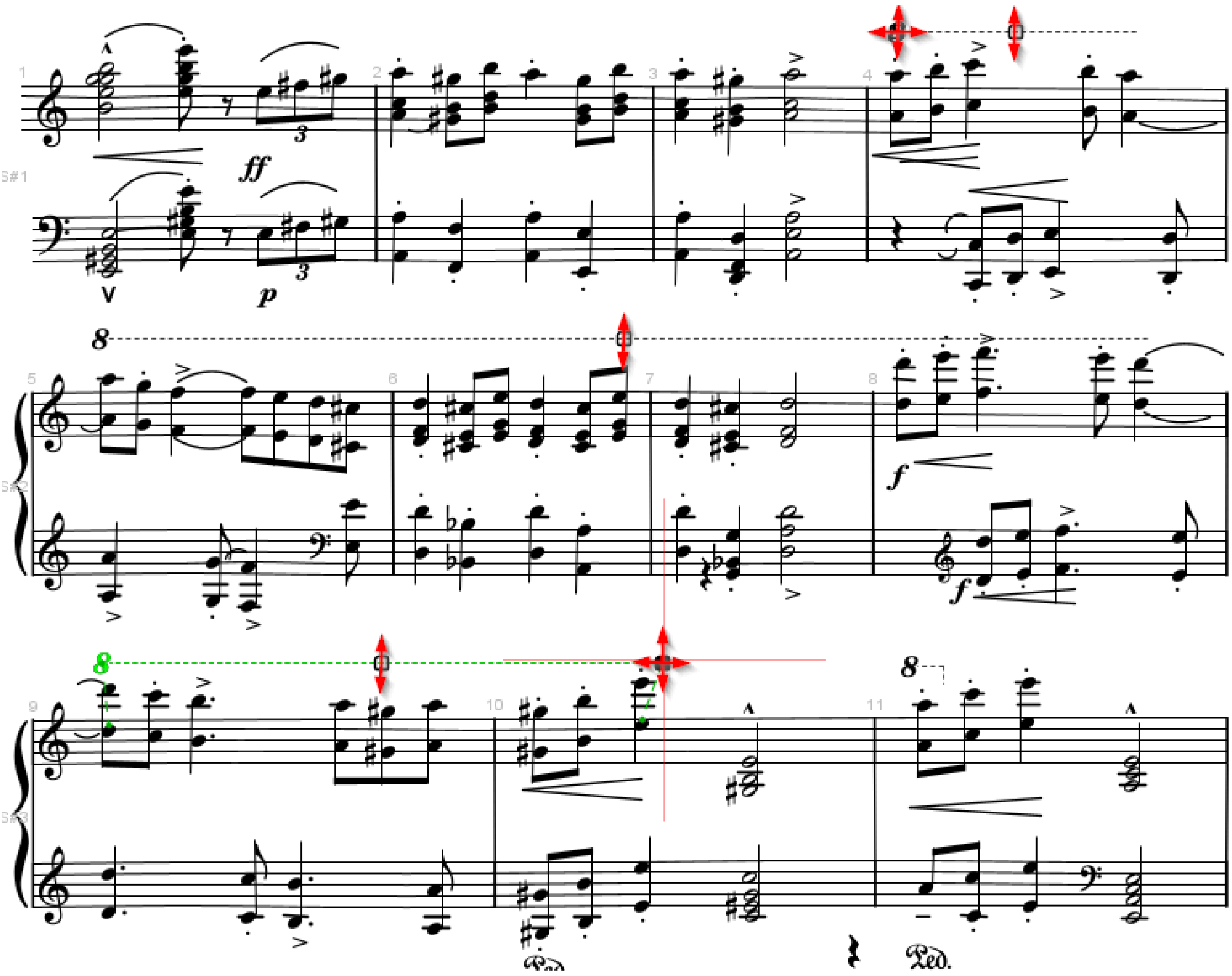
See details in Multi-measure rest section

Octave shift editor

Single-line editor:

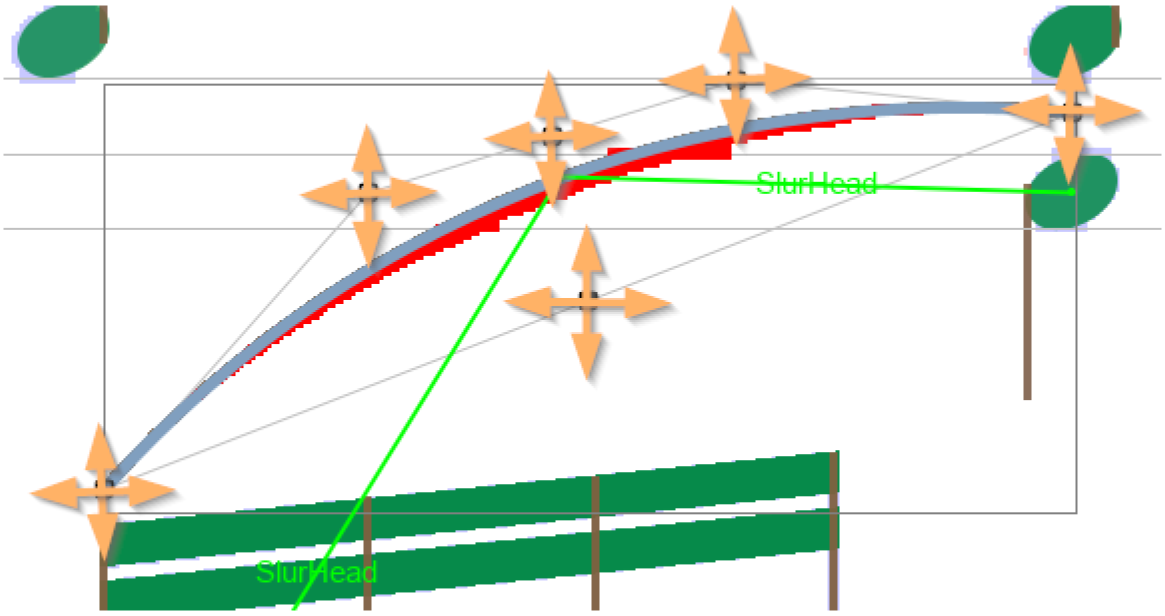


Multiple-line editor:



See details in Octave Shift section

Slur editor

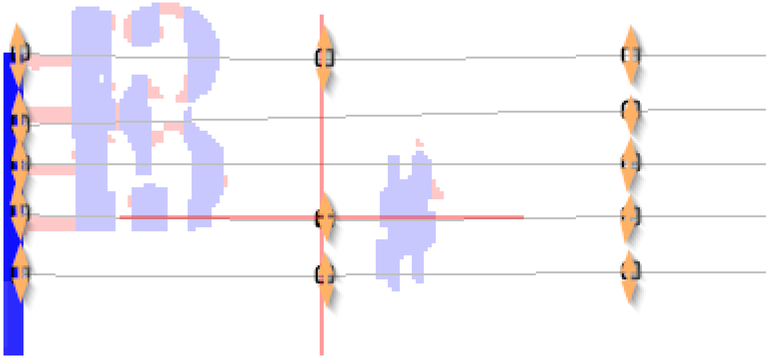


This is the most complex editor:

- Center handle shifts the whole slur in any direction,
- A side handle extends the slur side in any direction (together with the related control point),

- Control handles move their underlying control point,
- Middle of control segment moves both control points in any direction.

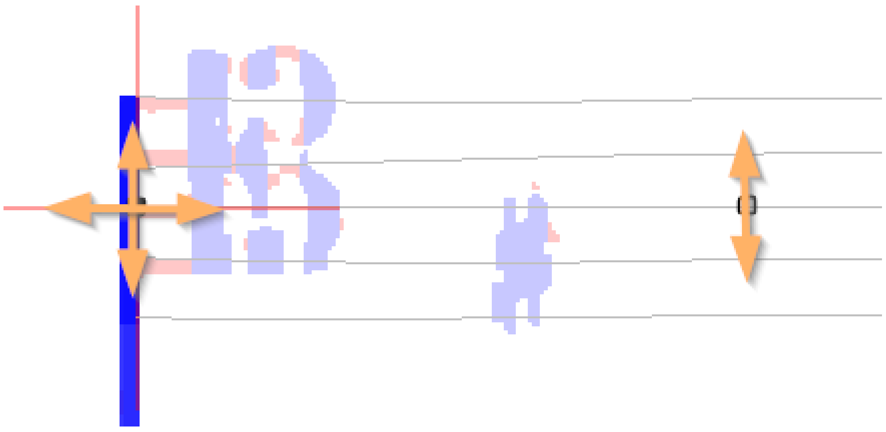
Staff lines editor



All the various lines handles in staff are available for individual vertical dragging.

See Staff Editing Lines mode.

Staff global editor

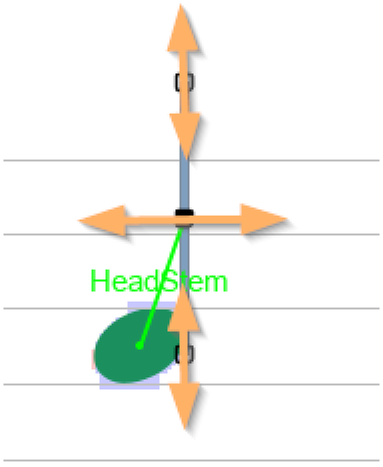


Handles are located on the staff middle line but they work for all lines as a whole (all lines are kept parallel when a handle is moved):

- Side handles (left and right) can be dragged both vertically and horizontally, thus allowing to stretch or shrink the staff.
  If the staff is stretched significantly, new defining points (and handles) are inserted. These inserted handles are meant to allow a fine vertical adjustment of the staff lines.
- Non-side handles can be dragged only vertically.

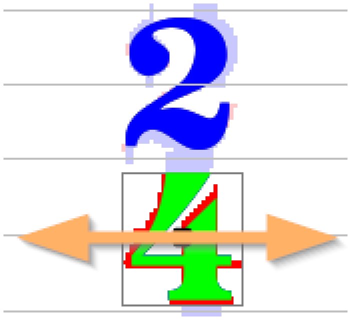
See Staff Editing Global mode.

Stem/Arpeggiato/Connector editor

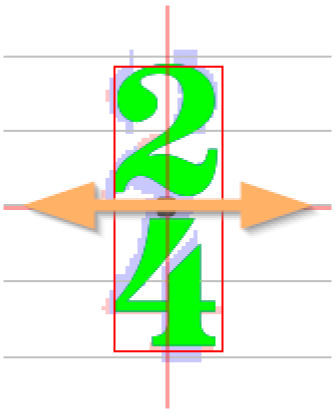


The editor applies to Stem as well as Arpeggiato and Connector (of barlines or brackets)

TimeHalf/Time editor



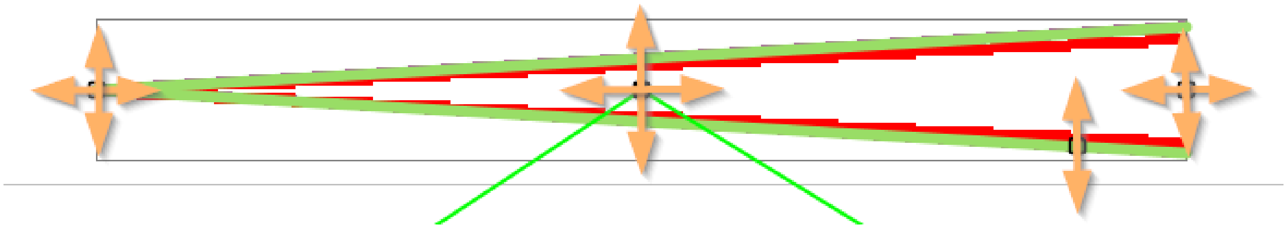
If the time signature was recognized with two separate halves, each of these halves is an inter by itself, and can be shifted horizontally only.



If the time signature is a whole signature then is can be shifted horizontally as a whole.

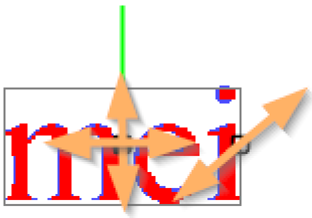
Note this applies as well to signatures handled globally, such as a manual custom signature or even a two-part signature (such as 2/4) if it was recognized globally.

Wedge editor



The handle on the lower segment allows to increase or decrease the wedge vertical spread.

Word editor



- Center handle shifts the word in any direction,
- Right handle modifies the word dimensions, by increasing or decreasing the font size.

Menus

This chapter simply describes all Audiveris pull-down and popup menus.

Since this chapter if meant for reference, each menu is described in its full extent, thus including advanced topics if any.
To be visible and callable, advanced topics must be explicitly selected via the Advanced Topics in the Preferences dialog.

TABLE OF CONTENTS

File menu

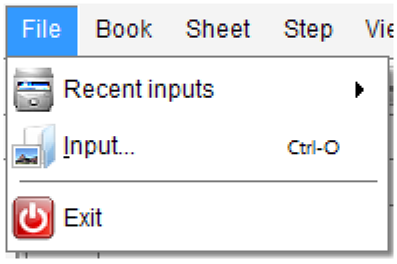


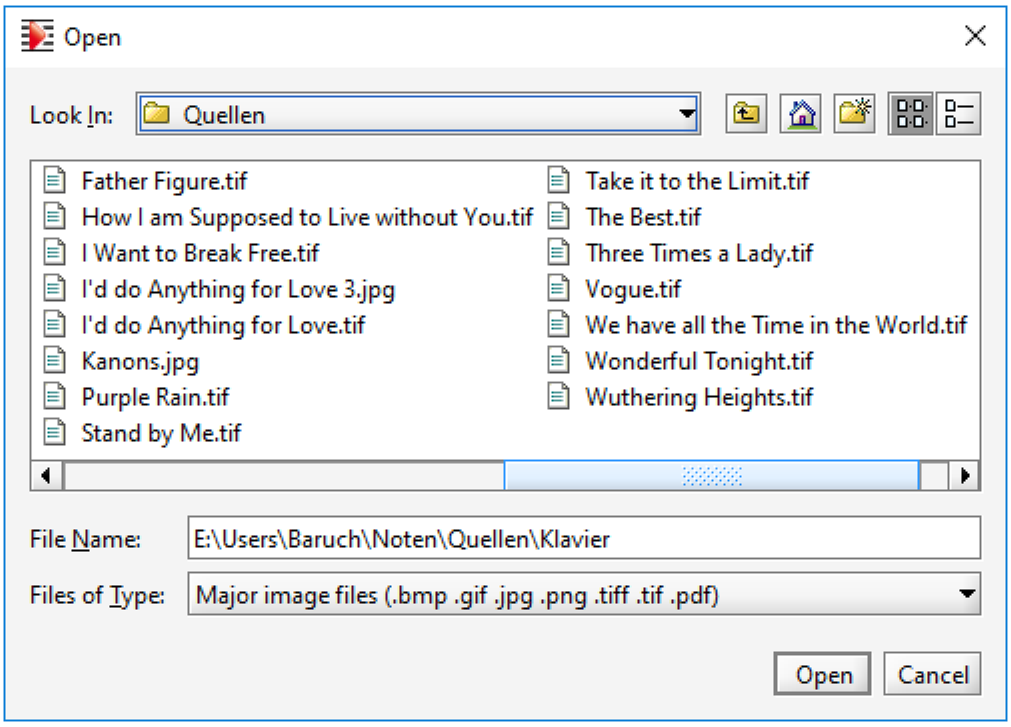
TABLE OF CONTENTS

Recent inputs

Gives a list of the most recent image input files. A click on the file name opens the file as source image.

Input

Opens a dialog box allowing to select the image or pdf-file that shall be transcribed. The dialog pre-selects the folder from where the last input image file was loaded.



Look In: Quellen

- Father Figure.tif
- How I am Supposed to Live without You.tif
- I Want to Break Free.tif
- I'd do Anything for Love 3.jpg
- I'd do Anything for Love.tif
- Kanons.jpg
- Purple Rain.tif
- Stand by Me.tif
- Take it to the Limit.tif
- The Best.tif
- Three Times a Lady.tif
- Vogue.tif
- We have all the Time in the World.tif
- Wonderful Tonight.tif
- Wuthering Heights.tif

File Name: E:\Users\Baruch\Noten\Quellen\Klavier

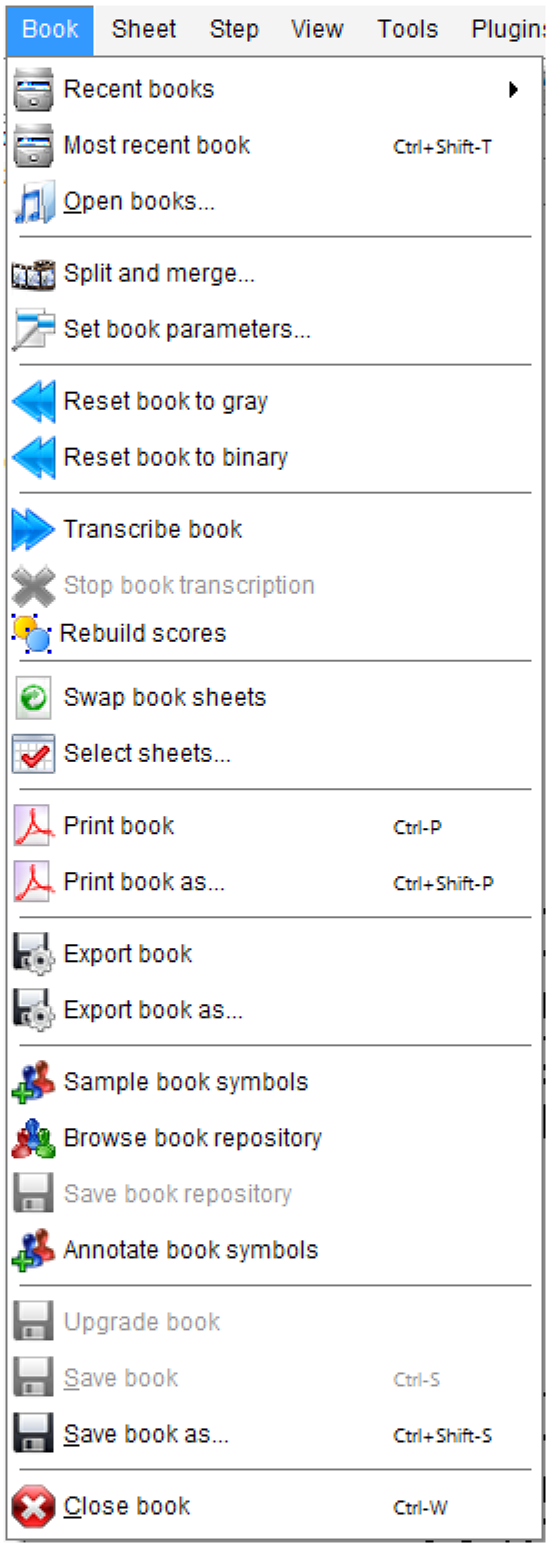
Files of Type: Major image files (.bmp .gif .jpg .png .tiff .tif .pdf)

Exit

Allows to exit Audiveris application in a civilized manner. You will first be prompted to save or discard any unsaved modification (book, sample repository).

NOTA: **Killing** the application via an external means does **not** save unsaved modifications, including modified options.

Book menu



A book is a collection of one or more sheets, each with its current state of transcription. It can be saved and reloaded at any time during a processing session.

As opposed to Loading an image which restarts from the initial image, loading a saved book allows to resume processing from the last step saved. This is especially useful for processing books with many sheets.

TABLE OF CONTENTS

Recent books

Open a list of recently saved books. After selection, the book is opened in the state it was saved.

Most recent book

Open the book most recently closed.

Open books

Open a dialog box allowing to select a book that has been saved before.

Split and merge

Open a dialog to select and combine inputs from different sources (books, image files).

See Split and merge section.

Set book parameters

Modify parameters for transcription, at global level, book level and sheets level.

See Book parameters.

Reset book to gray

Reset all sheets of a book to the initial state, as if it were just loaded as a plain image.

Reset book to binary

Same as `Reset book to bray` above, except that the initial processing of the images from gray-scale to B&W image is kept.

Transcribe book

Start the transcription of all sheets of the book.

Transcription is by default performed sheet by sheet. During processing, the user may start editing sheets that have already been transcribed.

Stop book transcription

Stop the current book transcription at the next step end.

Stop is achieved in a civilized manner, no data is corrupted. If desired, the processing can be relaunched manually.

Rebuild scores

Force the rebuilding of scores, out of the processed sheets.

Swap book sheets

Swap to disk all the sheets of current book (except the current sheet).

This is useful to save memory when working on a book with numerous sheets. When needed, a swapped sheet is transparently reloaded.

Select sheets

Select some sheets out of the current book, and limit processing to these sheets.

See Sheets Selection section.

Print book

Save the transcribed book in PDF format, so that it can be printed or saved for further purposes.

The name of the output file is derived from the book name. For instance, the printing of a `foo.pdf` input file will result in a `foo-print.pdf` file.

Print book as

Same as above, except that we can define the file name and the directory where the file is saved.

Export book

Export the transcribed score as a MusicXML `.mxl` file for exchange with a notation program. See Outputs Formats.

Export book as

Same as above except that we can define the file name and the directory where the file is saved.

Sample book symbols

Populate the book sample repository with samples derived from relevant inters of all book sheets.

(needs the `SAMPLES` topic, see Samples section)

Browse book repository

Open a window to browse, check and filter the samples of book repository.

(needs the `SAMPLES` topic, see Samples section)

Save book repository

Save book sample repository to disk.

(needs the `SAMPLES` topic, see Samples section)

Annotate book symbols

Populate a Zip archive with images and symbol annotations derived from book `Inter` instances.

These annotations are meant to populate a dataset for training the coming Audiveris 6.x new classifiers (Page and/or Patch).

(needs `ANNOTATIONS` topic)

Save book

Save the collection of all book sheets with the current state of transcription.

Save book as

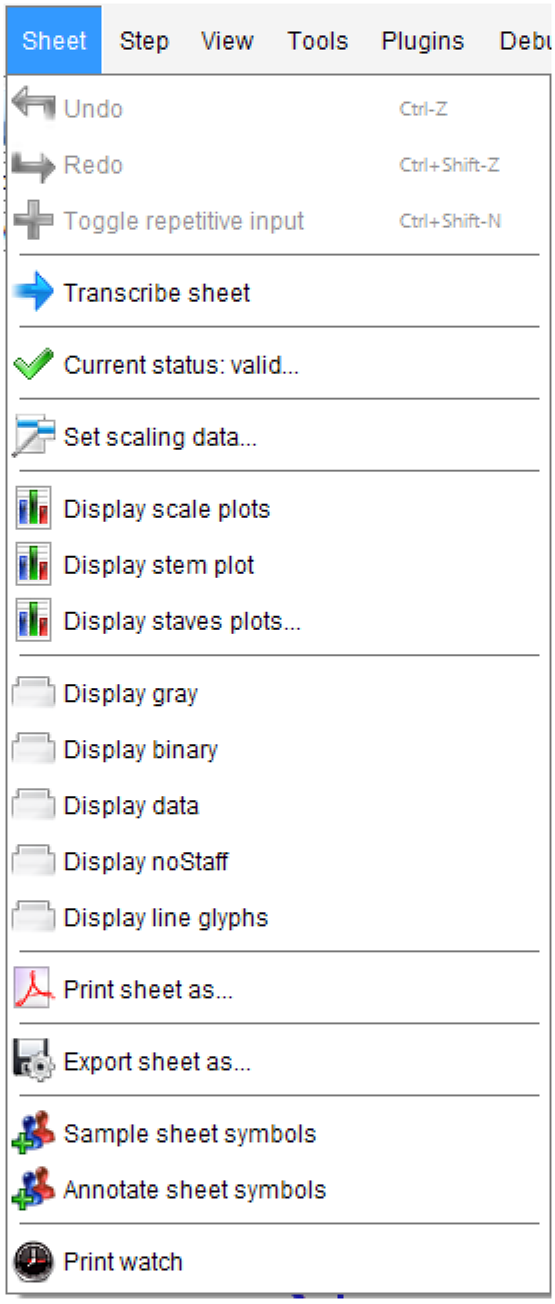
Same as above except that we can define the file name and the directory where the file is saved.

Close book

Close the current book and releases all its resources.

If the book has some unsaved data, the user is prompted to save it before closing.

Sheet menu



This menu operates on the current sheet (of the current book).

TABLE OF CONTENTS

5 Current status
6 Set scaling data
7 Display scale plots
8 Display stem plot
9 Display staves plots
10 Display gray
11 Display binary
12 Display data
13 Display noStaff
14 Display line glyphs
15 Print sheet as
16 Export sheet as
17 Sample sheet symbols
18 Annotate sheet symbols
19 Print watch

Undo

Undo last manual edit step, one at a time.

Redo

Redo last undone manual edit step, one at a time.

Toggle repetitive input

Switch on/off a mode where a simple click replicates the last inserted shape at the new mouse location.
This mode is convenient to insert a series of identical symbols, such as heads, at different locations.

Transcribe sheet

Start the transcription of just the active sheet.

Current status

Flag the current sheet as either valid or invalid (containing no music information, and thus be ignored during transcription of the book).

This validity status should not be confused with sheet selection made at book level:

- An **invalid** sheet contains no music and thus will always be ignored even if selected at book level.
- A sheet can be dynamically **selected** or not, according to user’s desire.

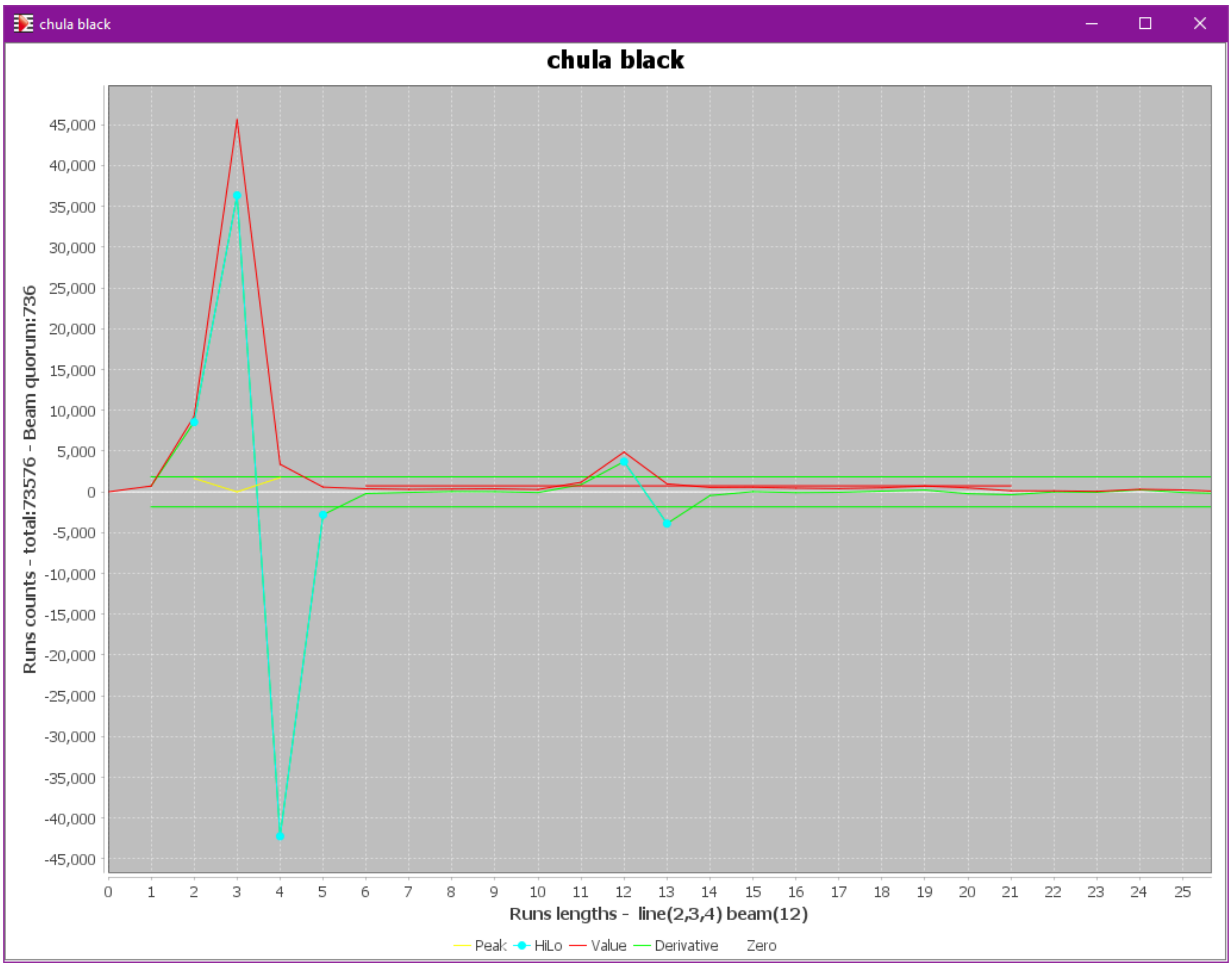
Set scaling data

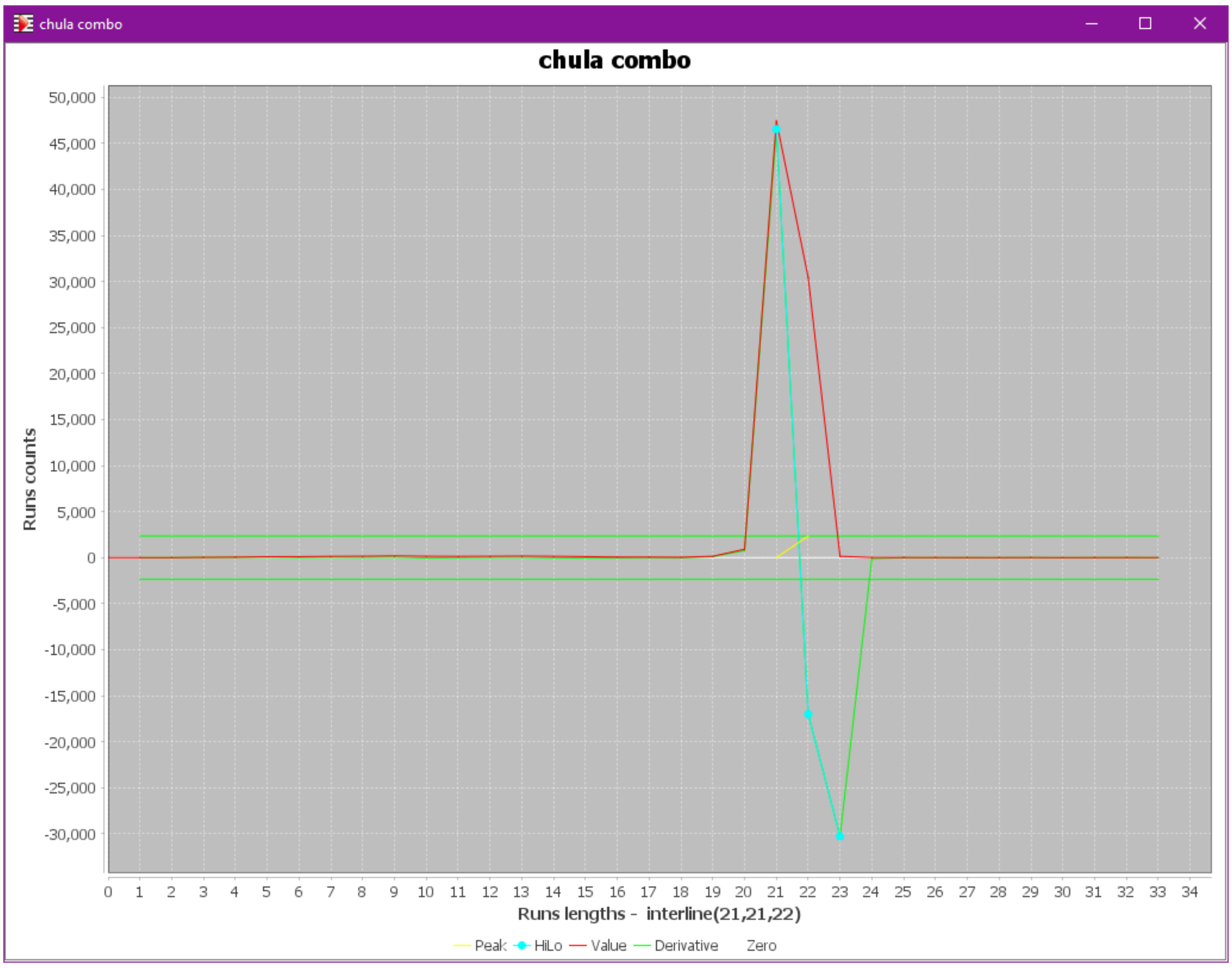
Display and potentially modify the scaling data of current sheet.

Display scale plots

Display plots of two histograms used for sheet scale retrieval:

- The “black” plot deals with vertical black runs, leading to staff line thickness and possibly beam thickness.
- The “combo” plot deals with combined length of vertical black run followed by vertical white run, leading to staff interline(s).

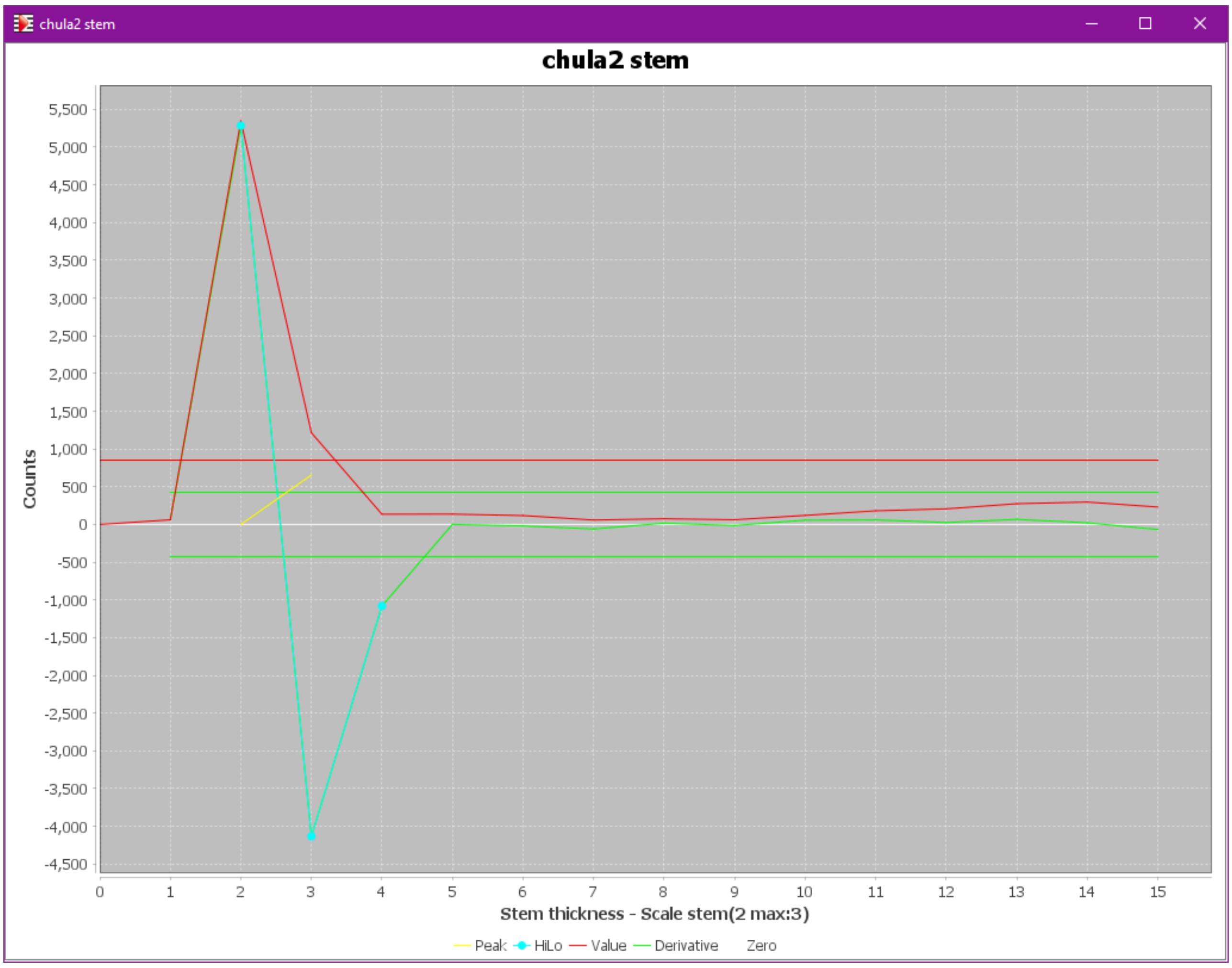




(needs `PLOTS` advanced topic)

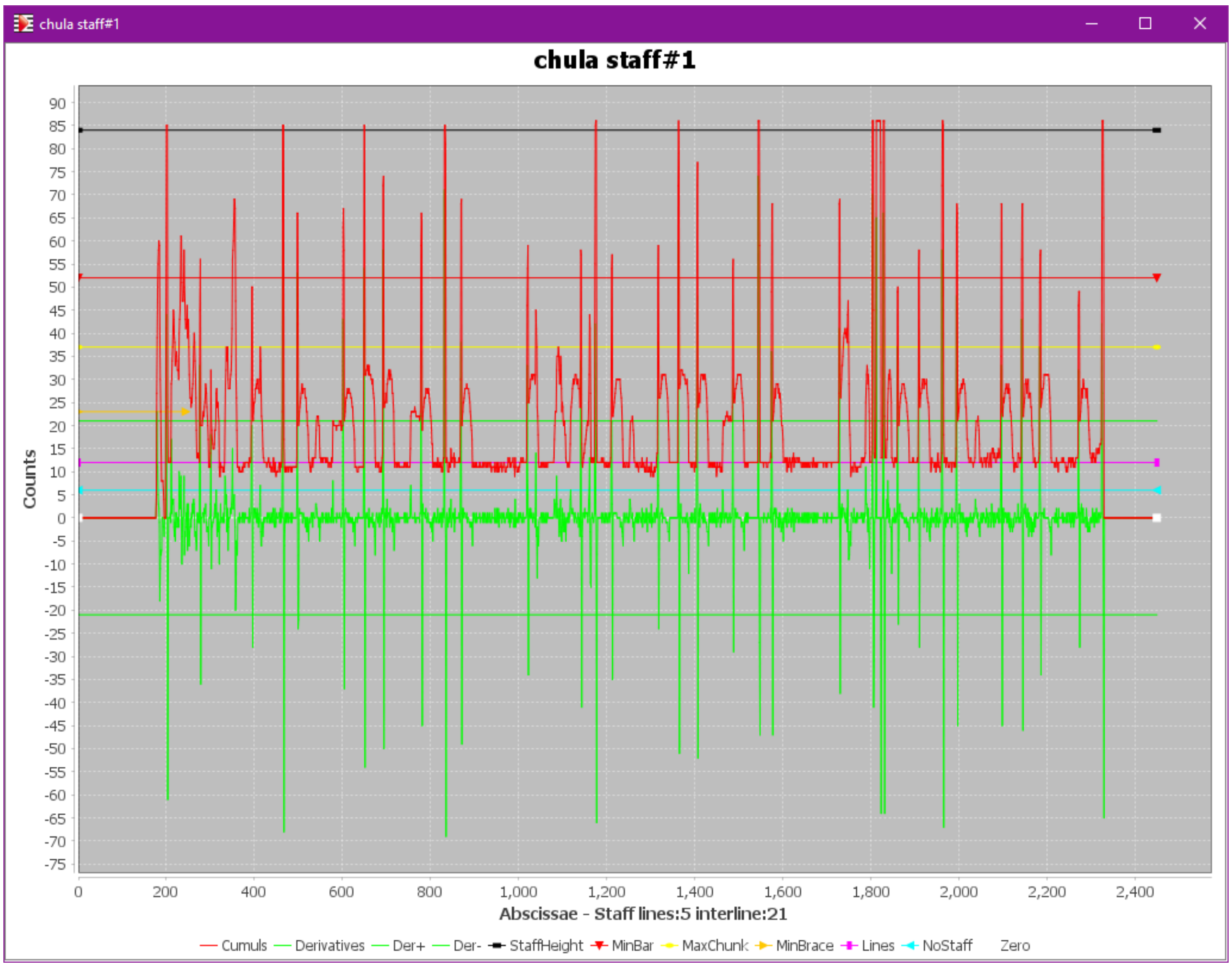
Display stem plot

Display plot of histogram of stem thickness.



(needs `PLOTS` advanced topic)

Display staves plots



Display the projection to x-axis of black pixels contained between staff first and last lines, leading to barlines extraction.

(needs `PLOTS` advanced topic)

Display gray

If still available, display the (initial) gray view tab.

Display binary

Open if needed, then select the binary view tab.

Display data

Select the data view tab. This central sheet view exists from GRID step onwards and is the only one which cannot be manually closed.

Display noStaff

Open if needed, then select the “NoStaff” view tab that shows the transcribed score with “logically erased” staff lines.

It’s helpful to see the quality of the image that is used for the internal score processing, because staff line removal can lead to collateral damages in some cases.

Display line glyphs

Open the “StaffLineGlyphs” view tab that shows just the “removed” pixels of staff lines.

Print sheet as

Write the transcribed sheet in PDF format, so that it can be printed or saved for further purposes.

The name of the output file is derived from the book name, followed by “#” and the sheet number if the book contains more than one sheet.

Export sheet as

Export the content of just the current sheet, using MusicXML format.

As for print, the name of the output file is derived from the book name and the sheet number.

Sample sheet symbols

Populate the book sample repository with samples derived from relevant inters of this sheet only.

(needs the `SAMPLES` advanced topic, see Samples section)

Annotate sheet symbols

Populate a Zip archive with images and symbol annotations derived from inters of this sheet only.

(needs the `ANNOTATIONS` advanced topic)

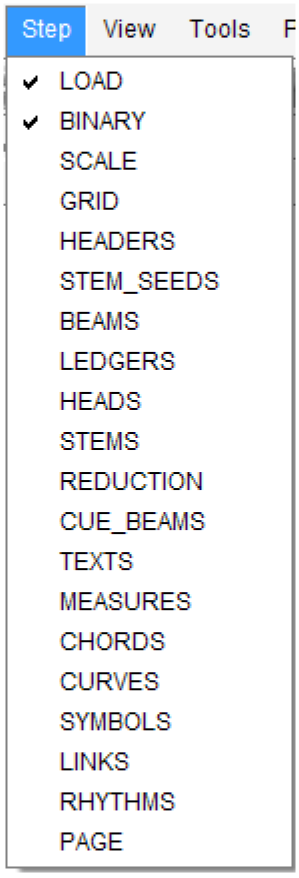
Print watch

On the console, print out the detailed durations of the steps run on the sheet.

```
StopWatch "chula"
-----------------------------------
   ms      % Task
-----------------------------------
  213   0.7% LOAD
  756   2.7% BINARY
  316   1.1% SCALE
 3615  12.7% GRID
 1855   6.5% HEADERS
  920   3.2% STEM_SEEDS
 5028  17.6% BEAMS
  518   1.8% LEDGERS
 6710  23.5% HEADS
 1851   6.5% STEMS
  321   1.1% REDUCTION
    1   0.0% CUE_BEAMS
 1719   6.0% TEXTS
  106   0.4% MEASURES
  442   1.6% CHORDS
 2193   7.7% CURVES
  715   2.5% SYMBOLS
  411   1.4% LINKS
  662   2.3% RHYTHMS
  159   0.6% PAGE
-----------------------------------
28511 100.0% Total
```

(needs the `SPECIFIC_ITEMS` advanced topic)

Step menu



The step menu allows to transcribe the selected sheet step by step.

This can help detect at what step a transcription problem occurs.

Reference / Menus / View menu

View menu

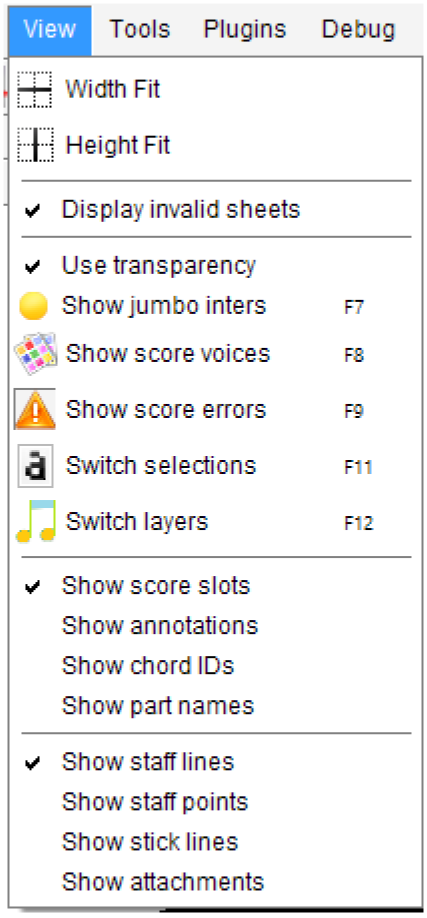


TABLE OF CONTENTS

Width fit

Zoom image to window width.

Height fit

Zoom image to window height.

Display invalid sheets

Toggle the display all sheets, even invalid ones.

Use transparency

Display interpretations in transparency according to the computed probability value.

Show jumbo inters (F7)

Toggle the display of certain shapes (dots by default) with a double size to ease their visual checking.

Show score voices (F8)

Toggle the display of notes and lyrics in colors according to the computed voices (only in output and mixed layer view).

Show score errors (F9)

Toggle the display of erroneous measures in pink background color.

Switch selections (F11)

Toggles the selection modes between glyph, inter and section.

(needs the `SPECIFIC_VIEWS` advanced topic)

Switch layers (F12)

Toggle between the 3 different layer views (see Sheet display modes).

Show score slots

Display the detected rhythm analysis (only in output and mixed view).

Show annotations

Toggle the display of all system, part, measure and rhythm related annotations (system ID, part name, measure ID, time slot offset).

Show chord IDs

Toggle the display of every chord ID.

This is convenient to visually map chords between measure strips and sheet view.

Show part names

Toggle the display of logical part name for every physical part.

Show staff lines

Toggle the display of staff lines.

(needs the `SPECIFIC_ITEMS` advanced topic)

Show staff points

Toggle the display of staff lines defining points, the ones that can be manually edited.

(needs the `SPECIFIC_ITEMS` advanced topic)

Show stick lines

Toggle the display of the average line of the selected glyph.

(needs the `SPECIFIC_ITEMS` advanced topic)

Show attachments

Toggle the display of specific attachments.

These attachments are drawings related to the processing of some entities (such as arc lookup areas to connect candidate slur portions). They are helpful only for visual checking of specific steps and thus are not persisted on disk.

(needs the `SPECIFIC_ITEMS` advanced topic)

Tools menu

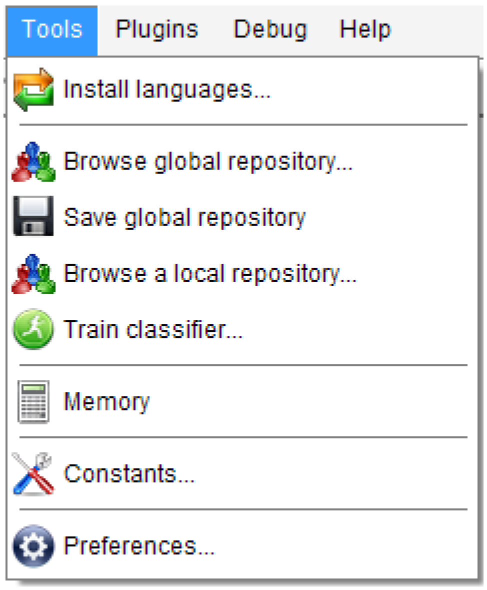


TABLE OF CONTENTS

Install languages

Open a dialog to select and download additional OCR languages. See OCR languages.

Browse global repository

Open a dialog to verify and edit the content of the Global sample repository (this is *the* only repository which is used for classifier training).

(needs the `SAMPLES` topic)

Save global repository

Saves the Global sample repository to disk.

(needs the `SAMPLES` topic)

Browse a local repository

Open a dialog to verify and edit a local (book) sample repository (generally before merging it into the Global repository).

(needs the `SAMPLES` topic)

Train classifier

Open a dialog to launch, monitor and evaluate the training of the glyph classifier.

See the Training section.

(needs the `SAMPLES` topic)

Memory

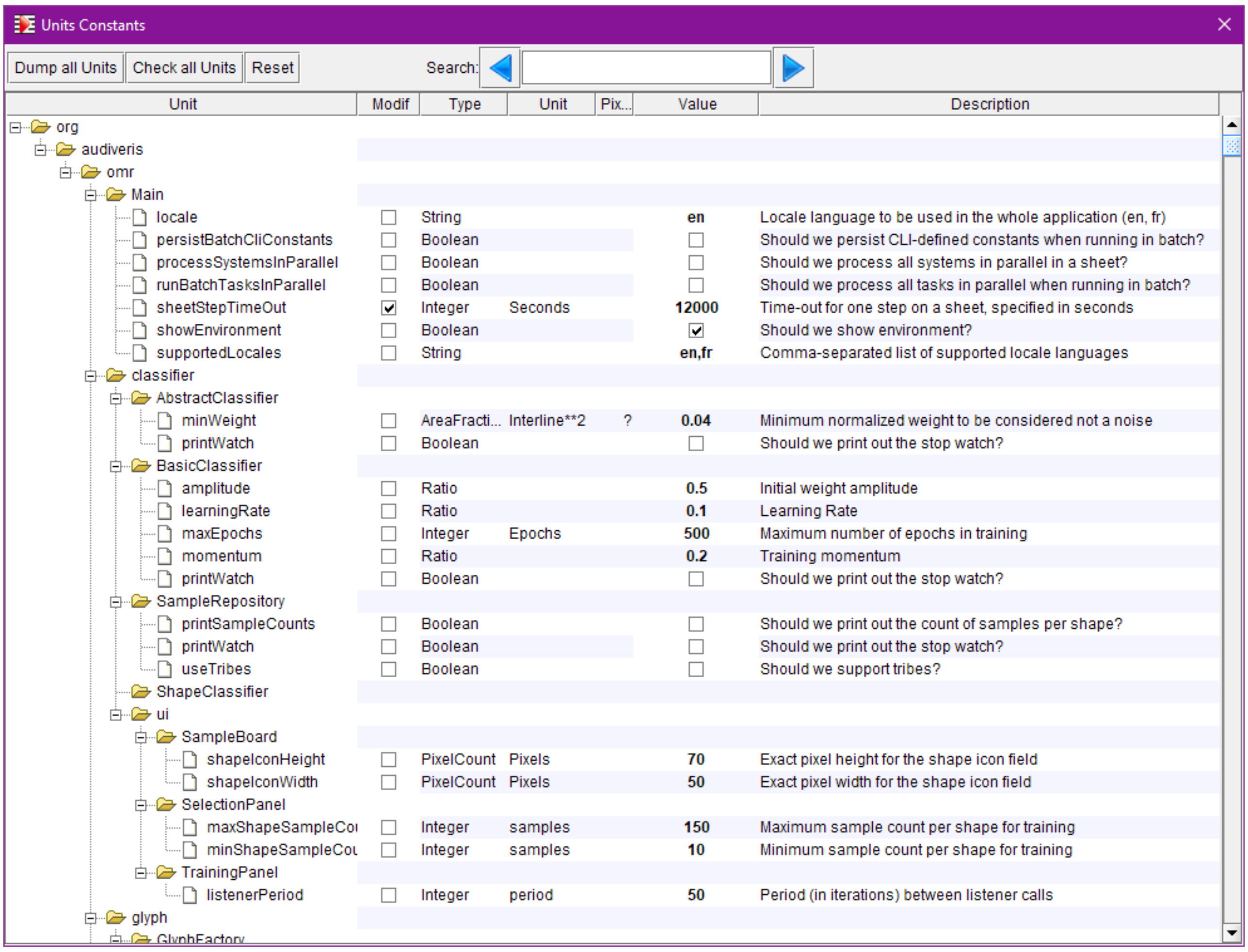
Displays the used memory in the output window

Constants

Opens the constants management window.

All available constants are listed. Most of them concern development constants only.

A search field allows to search for a string portion in the constant name or its description.



Unit	Modif	Type	Unit	Pix...	Value	Description
org						
audiveris						
omr						
Main						
locale	□	String			en	Locale language to be used in the whole application (en, fr)
persistBatchCliConstants	□	Boolean			□	Should we persist CLI-defined constants when running in batch?
processSystemsInParallel	□	Boolean			□	Should we process all systems in parallel in a sheet?
runBatchTasksInParallel	□	Boolean			□	Should we process all tasks in parallel when running in batch?
sheetStepTimeOut	☑	Integer	Seconds		12000	Time-out for one step on a sheet, specified in seconds
showEnvironment	□	Boolean			☑	Should we show environment?
supportedLocales	□	String			en,fr	Comma-separated list of supported locale languages
classifier						
AbstractClassifier						
minWeight	□	AreaFracti...	Interline**2	?	0.04	Minimum normalized weight to be considered not a noise
printWatch	□	Boolean			□	Should we print out the stop watch?
BasicClassifier						
amplitude	□	Ratio			0.5	Initial weight amplitude
learningRate	□	Ratio			0.1	Learning Rate
maxEpochs	□	Integer	Epochs		500	Maximum number of epochs in training
momentum	□	Ratio			0.2	Training momentum
printWatch	□	Boolean			□	Should we print out the stop watch?
SampleRepository						
printSampleCounts	□	Boolean			□	Should we print out the count of samples per shape?
printWatch	□	Boolean			□	Should we print out the stop watch?
useTribes	□	Boolean			□	Should we support tribes?
ShapeClassifier						
ui						
SampleBoard						
shapeIconHeight	□	PixelCount	Pixels		70	Exact pixel height for the shape icon field
shapeIconWidth	□	PixelCount	Pixels		50	Exact pixel width for the shape icon field
SelectionPanel						
maxShapeSampleCo	□	Integer	samples		150	Maximum sample count per shape for training
minShapeSampleCou	□	Integer	samples		10	Minimum sample count per shape for training
TrainingPanel						
listenerPeriod	□	Integer	period		50	Period (in iterations) between listener calls
glyph						
GlyphFactory						

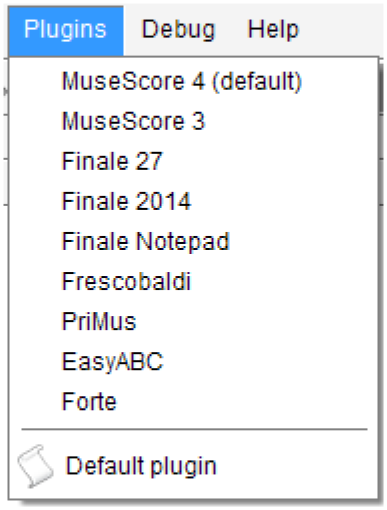
See the Constants section.

Preferences

Opens a dialog where the users can make their own choices regarding: the default processing of input images, the default plugin, the policy for outputs location, the font size in application items, the chosen locale, and additional items in pull-down menus.

See the Preferences section.

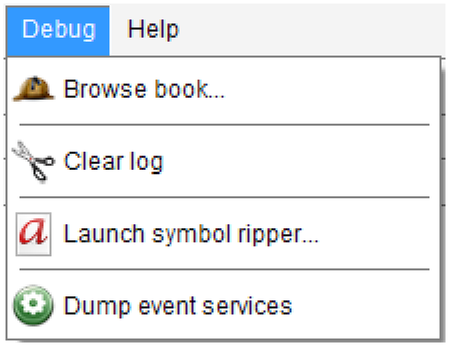
Plugins menu



Display the registered plugins to external programs.

See the Plugins section.

Debug menu

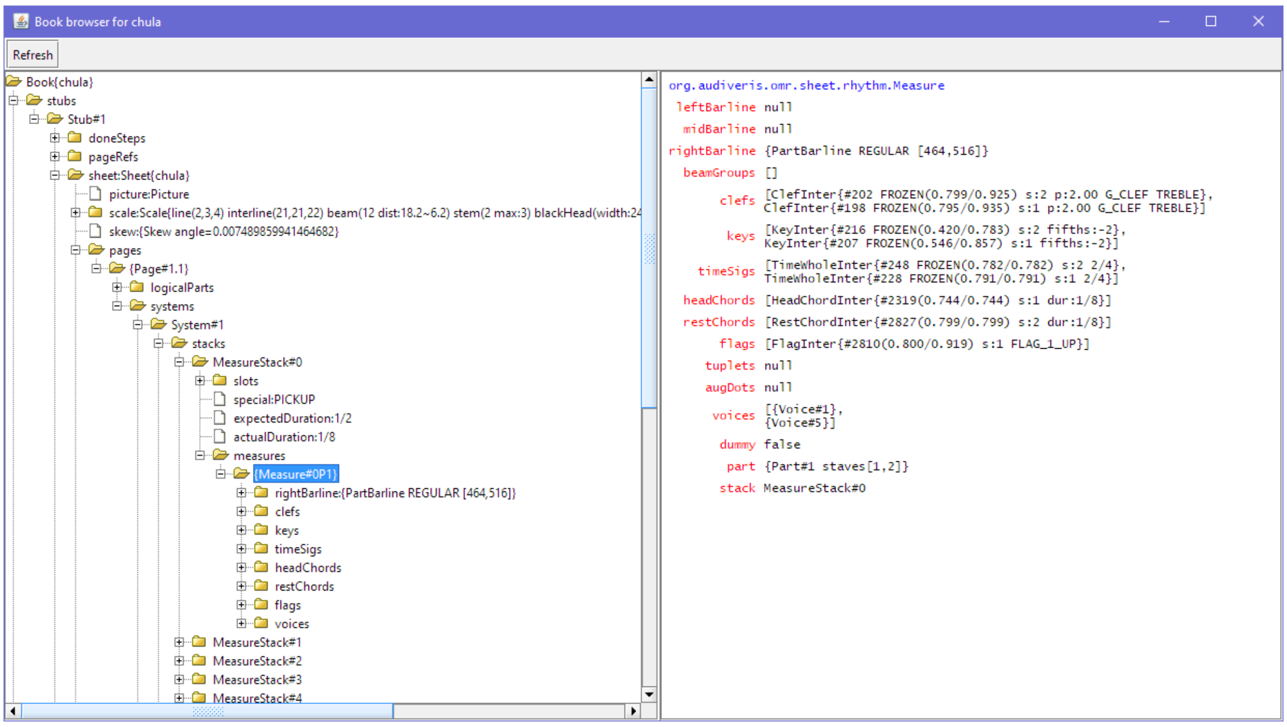


This menu makes a few debug actions conveniently available.

> The casual user can safely ignore this menu. To make it visible, you must have beforehand selected the `DEBUG` topic in the Advanced Topics of the Preferences dialog.

TABLE OF CONTENTS

Browse book



```
Book{chula}
  stubs
    Stub#1
      doneSteps
      pageRefs
      sheet:Sheet{chula}
        picture:Picture
        scale:Scale{line(2,3,4) interline(21,21,22) beam(12 dist:18.2~6.2) stem(2 max:3) blackHead(width:24
        skew:{Skew angle=0.007489859941464682}
        pages
          {Page#1.1}
            logicalParts
            systems
              System#1
                stacks
                  MeasureStack#0
                    slots
                    special:PICKUP
                    expectedDuration:1/2
                    actualDuration:1/8
                    measures
                      {Measure#0P1}
                        rightBarline:{PartBarline REGULAR [464,516]}
                        clefs
                        keys
                        timeSigs
                        headChords
                        restChords
                        flags
                        voices
                  MeasureStack#1
                  MeasureStack#2
                  MeasureStack#3
                  MeasureStack#4
```

```
org.audiveris.omr.sheet.rhythm.Measure
 leftBarline null
  midBarline null
rightBarline {PartBarline REGULAR [464,516]}
  beamGroups []
       clefs [ClefInter{#202 FROZEN(0.799/0.925) s:2 p:2.00 G_CLEF TREBLE},
             ClefInter{#198 FROZEN(0.795/0.935) s:1 p:2.00 G_CLEF TREBLE}]
        keys [KeyInter{#216 FROZEN(0.420/0.783) s:2 fifths:-2},
             KeyInter{#207 FROZEN(0.546/0.857) s:1 fifths:-2}]
    timeSigs [TimeWholeInter{#248 FROZEN(0.782/0.782) s:2 2/4},
             TimeWholeInter{#228 FROZEN(0.791/0.791) s:1 2/4}]
  headChords [HeadChordInter{#2319(0.744/0.744) s:1 dur:1/8}]
  restChords [RestChordInter{#2827(0.799/0.799) s:2 dur:1/8}]
       flags [FlagInter{#2810(0.800/0.919) s:1 FLAG_1_UP}]
     tuplets null
     augDots null
      voices [{Voice#1},
             {Voice#5}]
       dummy false
        part {Part#1 staves[1,2]}
       stack MeasureStack#0
```

This dialog allows the user to browse the whole hierarchical content of the current book.

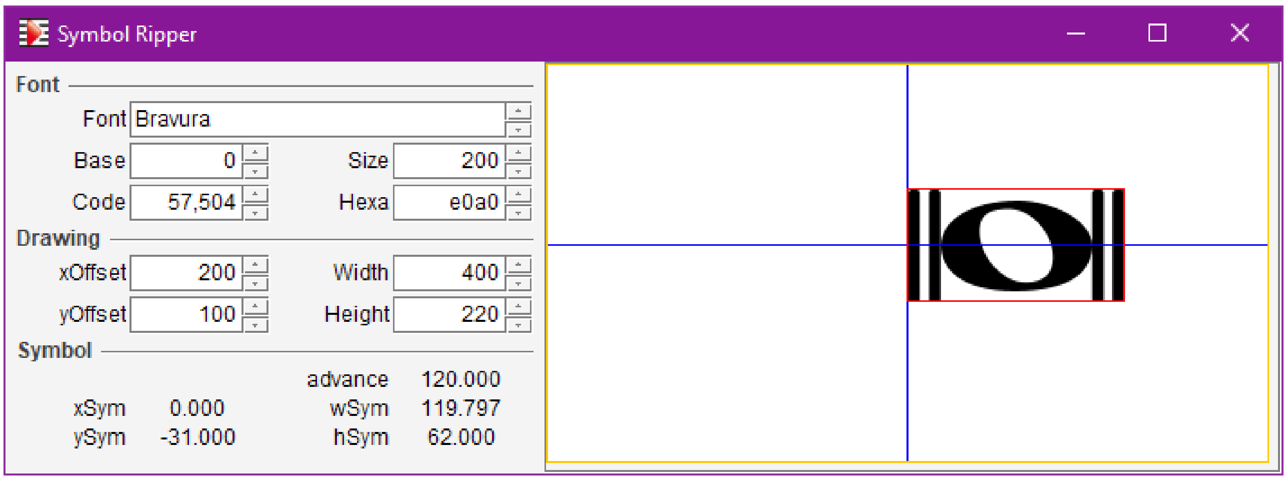
- The left part is a tree meant to navigate through the book hierarchy of sheets, pages, systems, measures, chords, glyphs, inters, etc.
- The right part displays the name and value of each data member of the entity currently selected in the tree.

Clear log

This action clears the events panel (located at bottom of main window). This is meant for an easier focus on the coming events.

Note this does not impact the output, nor the log file.

Launch symbol ripper



Font

Font: Bravura

Base: 0 Size: 200

Code: 57,504 Hexa: e0a0

Drawing

xOffset: 200 Width: 400

yOffset: 100 Height: 220

Symbol

advance 120.000

xSym 0.000 wSym 119.797

ySym -31.000 hSym 62.000

This action launches the “Symbol Ripper” dialog which is meant to study symbols in different fonts.

- The “Font” field allows to select a font among the installed ones on your machine.
- The symbol code can be entered in the “Code” field (decimal value) or in the “Hexa” field (hexadecimal value)
- The “Base” field can define an offset to be added to the entered code/hexa value.
- The “Size” field defines the point size for the selected font.

The other fields display the symbol physical dimensions (advance, x, y, width, height) in the selected size.

In the example above, we are on the “Bravura” font, where the hexadecimal code “E0A0” refers to the BREVE symbol.

Dump event services

Dump all UI services connected to the current sheet.

A UI service is an instance of EventBus, with publish/subscribe features, which allows to decouple UI agents.

- Typically, a glyph board will be subscribed to the bus in charge of glyph selection, in order to display the glyph details as soon as a glyph is published on the glyph bus by some agent.
- The same glyph board also allows the user to type a glyph ID, which will be published on the glyph ID bus, and the glyph index (subscribed on the glyph ID bus) will search its index with the ID and publish the corresponding glyph on the glyph bus, which will be picked up by the glyph board, among other

subscribers.

For each allocated UI service (such as locationService, glyphIndexService, interIndexService), this action lists:

- Each event class handled by the service,
- The latest instance, if any, published in the event class,
- Every agent (address and name) subscribed to this event class.

Here is an example:

```
Selection services of Stub#1
locationService subscribers:
   LocationEvent: 6 LocationEvent{ src:SymbolsEditor-MyView ENTITY_INIT java.awt.Rectangle[x=2254,y
      @3a261e6f glyphIndexService size:973
      @2d727d16 hLagService size:2236
      @6f854dad vLagService size:3252
      @1ae6e8ef interIndexService size:771
      @22241722 SymbolsEditor-MyView
      @336ce775 filamentIndexService size:0
No pixelService
hLagService subscribers:
   IdEvent: 1
      @2d727d16 hLagService size:2236
   EntityListEvent: 2
      @2d727d16 hLagService size:2236
      @22241722 SymbolsEditor-MyView
vLagService subscribers:
   IdEvent: 1
      @6f854dad vLagService size:3252
   EntityListEvent: 2
      @6f854dad vLagService size:3252
      @22241722 SymbolsEditor-MyView
filamentIndexService subscribers:
   EntityListEvent: 1 EntityListEvent{ src:filamentIndexService PRESSING []}
      @336ce775 filamentIndexService size:0
   IdEvent: 1
      @336ce775 filamentIndexService size:0
glyphIndexService subscribers:
   IdEvent: 1 IdEvent{ src:{SymbolGlyphBoard Glyph} ENTITY_INIT 2793}
      @3a261e6f glyphIndexService size:973
   EntityListEvent: 4 EntityListEvent{ src:glyphIndexService ENTITY_INIT [BasicGlyph{#2793 [SYMBOL]
      @3a261e6f glyphIndexService size:973
      @22241722 SymbolsEditor-MyView
      @28388473 {EvaluationBoard Basic Classifier}
      @3321ce77 {SymbolGlyphBoard Glyph}
interIndexService subscribers:
   EntityListEvent: 3 EntityListEvent{ src:interIndexService PRESSING []}
      @1ae6e8ef interIndexService size:771
      @22241722 SymbolsEditor-MyView
      @220219b7 {InterBoard `Inter`}
   IdEvent: 1
      @1ae6e8ef interIndexService size:771
```

Help menu

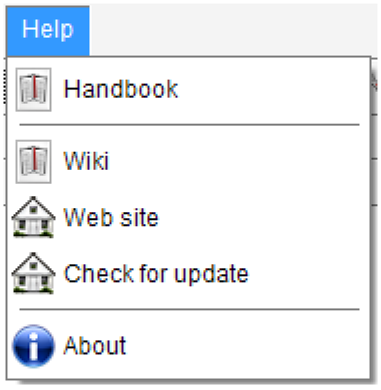


TABLE OF CONTENTS

Handbook

Launch a browser of this handbook documentation.

Wiki

Launch a browser on the Audiveris Wiki.

Web site

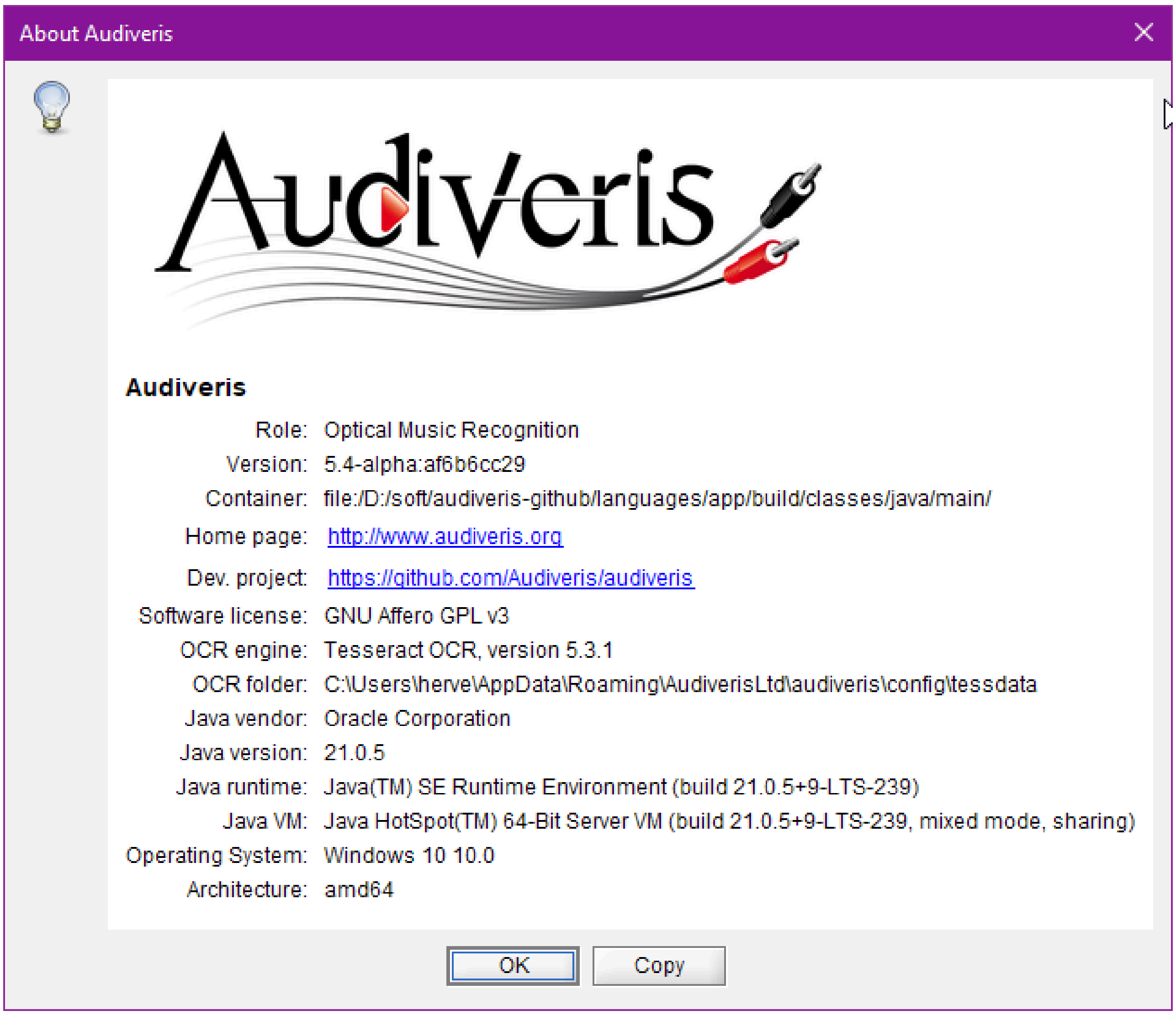
Launch a browser on the GitHub Audiveris organization and its contained repositories.

Check for update

Check the GitHub site for a newer release of Audiveris.

About

Open an “a propos” dialog with information about Audiveris program and the main external components it depends upon (Tesseract OCR, Java, OS, machine architecture).



Audiveris

Role: Optical Music Recognition
Version: 5.4-alpha:af6b6cc29
Container: file:/D:/soft/audiveris-github/languages/app/build/classes/java/main/
Home page: http://www.audiveris.org
Dev. project: https://github.com/Audiveris/audiveris
Software license: GNU Affero GPL v3
OCR engine: Tesseract OCR, version 5.3.1
OCR folder: C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\config\tessdata
Java vendor: Oracle Corporation
Java version: 21.0.5
Java runtime: Java(TM) SE Runtime Environment (build 21.0.5+9-LTS-239)
Java VM: Java HotSpot(TM) 64-Bit Server VM (build 21.0.5+9-LTS-239, mixed mode, sharing)
Operating System: Windows 10 10.0
Architecture: amd64

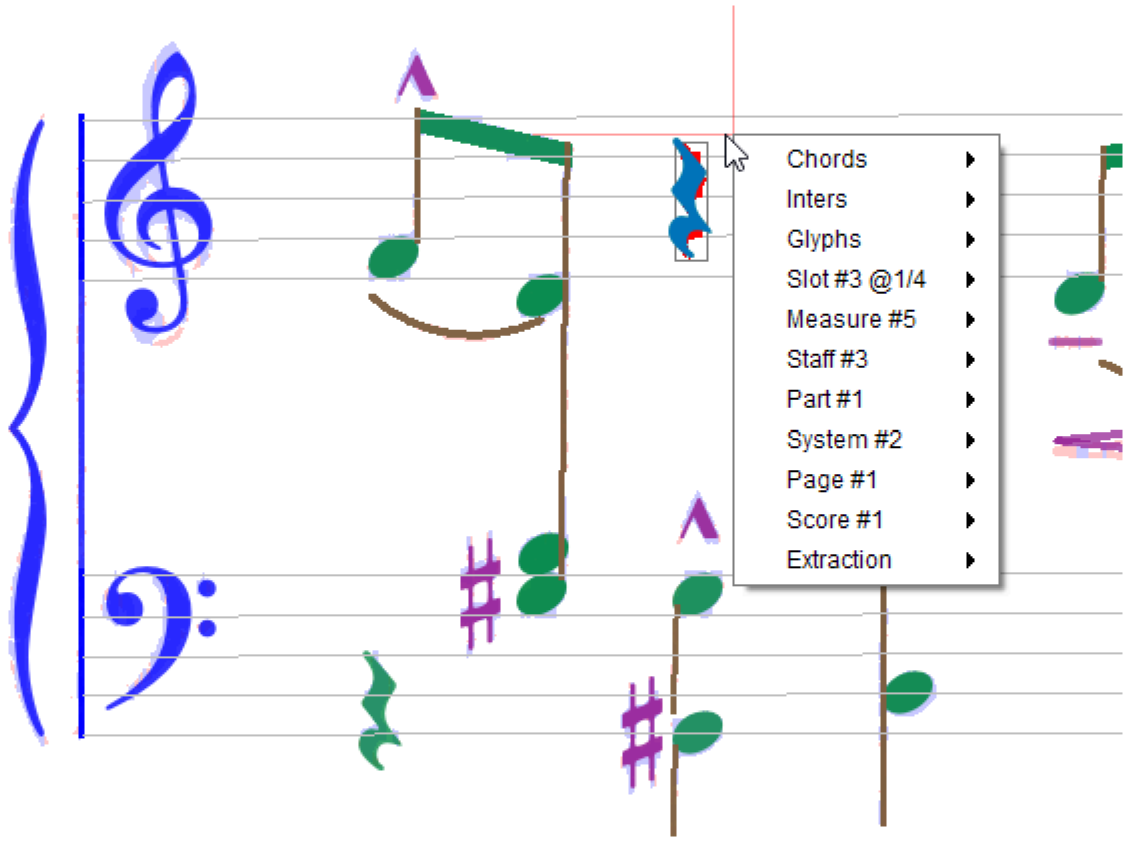
Since 5.4, there is a `Copy` button. It copies the textual content of this dialog to the clipboard, to ease the sharing of this information on Audiveris forums (issues or discussions).

This gives, for example:

```
Audiveris
Role: Optical Music Recognition
Version: 5.4-alpha:af6b6cc29
Container: file:/D:/soft/audiveris-github/languages/app/build/classes/java/main/
```

```
Home page: http://www.audiveris.org

Dev. project: https://github.com/Audiveris/audiveris

Software license: GNU Affero GPL v3

OCR engine: Tesseract OCR, version 5.3.1

OCR folder: C:\Users\herve\AppData\Roaming\AudiverisLtd\audiveris\config\tessdata

Java vendor: Oracle Corporation

Java version: 21.0.5

Java runtime: Java(TM) SE Runtime Environment (build 21.0.5+9-LTS-239)

Java VM: Java HotSpot(TM) 64-Bit Server VM (build 21.0.5+9-LTS-239, mixed mode, sharing)

Operating System: Windows 10 10.0

Architecture: amd64
```

Pop-up menu



This pop-up menu appears via a mouse right-click in a sheet view, and is represented by the ≡ symbol in this handbook.

It displays *contextual* information and actions that depend on the sheet latest processing step, the current location in sheet view and the selected items if any.

Some of its content also depends on user-selected Advanced Topics in the Preferences dialog.

TABLE OF CONTENTS

Chords

Depending on the chord(s) currently selected, this sub-menu allows to modify their configuration regarding time slot and voice.

See the chapter on Chords editing.

Inters

If at least one `Inter` instance has been selected, this sub-menu lists the selected Inters, ordered by decreasing grade.

It also allows to delete `Inter` instance(s) or relation(s).

Glyphs

If at least one Glyph instance has been selected, the *compound* glyph (dynamically built by merging all selected glyphs) appears in this sub-menu.

It allows to assign a specific shape (via a created `Inter`) to the selected glyph.

Slot #n @offset

If the `RHYTHMS` step has been reached, this sub-menu relates to the closest time-slot in the containing measure stack.

For this time slot, we can:

- **Dump chords**: list all the chords starting on this slot.
- **Dump voices**: list all the voices with chords starting on this slot.

Measure #n

If current location lies within a measure, this sub-menu provides actions upon the selected measure.

A measure is delimited horizontally by the left and right bar-lines and vertically by the top and bottom staves of the containing part.

Depending on which steps have already been performed, we can:

- **Dump stack voices**: for the containing vertical stack of measures, display a kind of strip with time slots in abscissa and voices in ordinate.
- **Dump measure voices**: same display but limited to the containing measure.
- **Reprocess rhythm**: force re-computation of rhythm data (slots and voices) in the current measure.
- **Merge on right**: merge the current measure with the next measure on right.

Example of voices dump:

```
MeasureStack#2
    |0       |1/16    |3/16    |1/4     |3/8     |7/16    |1/2
--- P1
V 1 |Ch#2325 ==================|Ch#2326 |Ch#2828 |Ch#2327 |1/2
V 5 |Ch#2347 |Ch#2348 |Ch#2349 |Ch#2350 |Ch#2351 =========|1/2
```

Staff #n

This sub-menu appears if the current location is within a staff height (even beyond staff horizontal limits).

We can edit the whole staff or one of its lines (see Staff Editing):

- **Edit staff**: Manual editing of all staff lines as a whole
- **Edit lines**: Manual editing of one staff line in isolation

We can also display vertical projections of the whole staff and of the staff header, provided that the `PLOTS` advanced topic has been selected.

Part #n

Allows to manually assign a logical part to this physical part.

See the section on part Manual mapping.

System #n

If the current system is not the last one in sheet, this allows to **Merge with system below**.

Page #n

Allows to **Reprocess rhythm**, that is to force re-computation of rhythm data for the whole page.

Score #n

Allows to manage all the logical parts in the containing score.
See the chapter on logical part management

Extraction

Extracts a rectangular portion (or whole) of the underlying binary image and save it to disk. This is meant for sharing or further analysis.

Limitation: The rectangular area selection is effective only from the `Binary` tab, not from the `Data` tab.

Boards

This chapter gathers the descriptions of all boards.

A board is a small UI panel, generally located in the right column of Audiveris main window (the boards column), although it can also appear in other windows such as the sample repository browser.

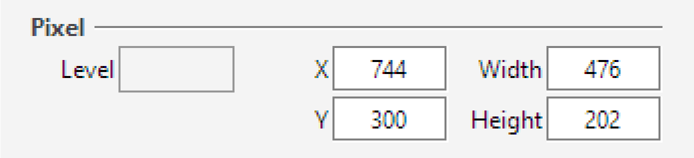
It is focused on a specific kind of OMR entity (pixel, glyph, inter, section, sample, …) and offers input and output features related to the OMR entity kind.

A board reads and/or writes to one or several event buses according to the OMR entity kind handled. See Debug menu for further details on event services.

Most boards can be dynamically shown or hidden via a right-click in the boards column.

TABLE OF CONTENTS

Pixel board



Pixel

Level		X	744	Width	476
		Y	300	Height	202

The pixel board handles the location within the sheet image, as well as the pixel gray value at the (x,y) location.

TABLE OF CONTENTS

Level

(output)
Gray value, between 0 and 255, at the (x,y) location.

This field is active only on the initial (gray) image. It becomes inactive on the binary (black & white) image.

X & Y

(input/output)
The abscissa and ordinate values for the selected location (either a single point, or the top left corner of a rectangle).

Width & Height

(input/output)
The width and height values of the selected rectangle. If both are zero, the

rectangle degenerates as a single point.

Reference / Boards / Binarization board

Binarization board

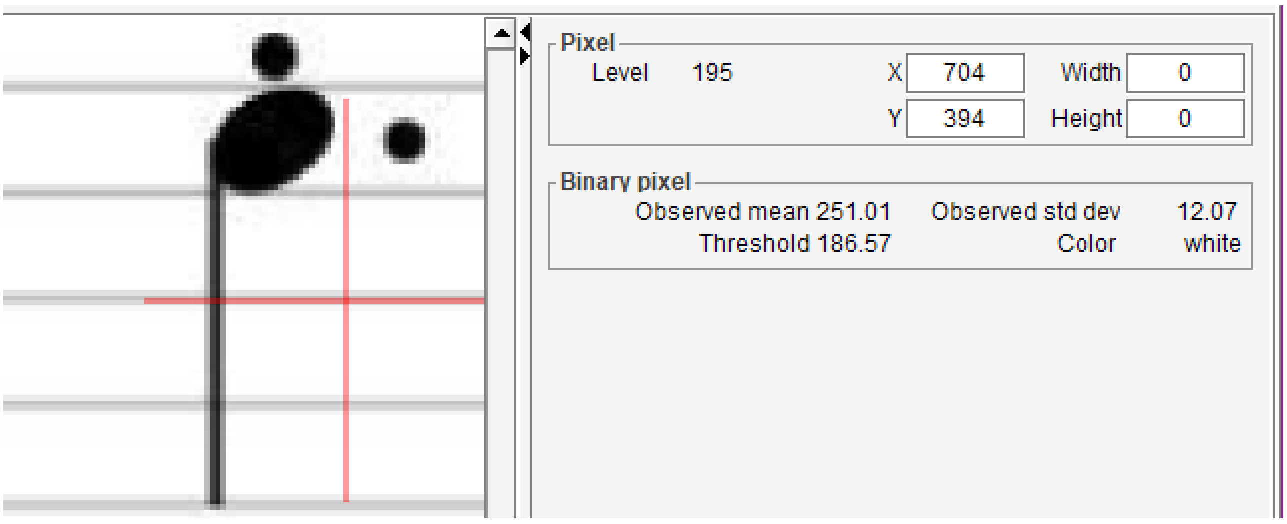
In release 5.4, the new binarization board provides the tools to manually select the binarization filter (either global or adaptive), adjust its settings and immediately see their impact on the sheet image.

This new board is available only in the Binary tab, and is *not* selected by default.

TABLE OF CONTENTS

Board in action

Let us take a gray image as input, here seen from the Gray tab:



Pixel

Level 195

X 704

Y 394

Width 0

Height 0

Binary pixel

Observed mean 251.01

Observed std dev 12.07

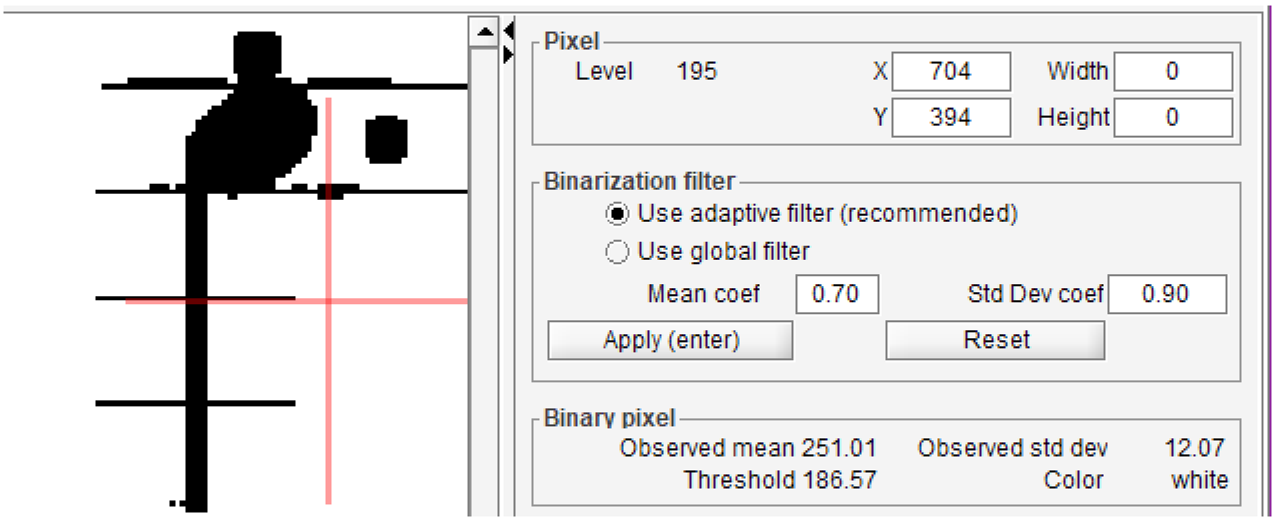
Threshold 186.57

Color white

Default adaptive filter

In the Binary tab on the same image, we can see both the Binarization board and the Binary board.

The sheet view displays the result of applying the default binarization filter.
In this specific example, we can observe that many lines portions are missing:



Pixel

Level 195 X 704 Width 0

Y 394 Height 0

Binarization filter

● Use adaptive filter (recommended)

○ Use global filter

Mean coef 0.70 Std Dev coef 0.90

Binary pixel

Observed mean 251.01 Observed std dev 12.07

Threshold 186.57 Color white

In the Binarization board, we can:

1. Select the filtering mode – either adaptive (the default) or global
2. Modify the settings of the filter
   - The threshold value for the global filter
   - The coefficients for mean and standard deviation values for the adaptive filter, from which the local threshold values will be computed

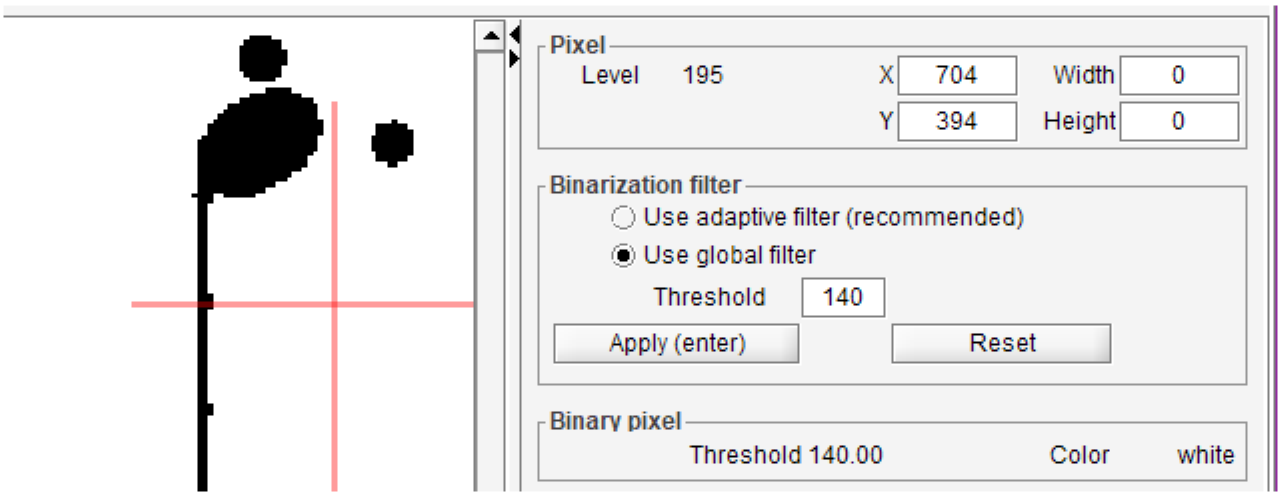
For the vast majority of input images, the adaptive filter gives the best results.

But for the image used in this example, we can try to modify the settings of the adaptive filter, the results will never be OK. The reason is the staff lines in this particular image use a pale gray color.
A global filter might better work.

Default global filter

Here, we have switched to the global filter, using its default threshold value (140).

The results are even worse, all the staff lines have disappeared:



Pixel
Level 195 X 704 Width 0
Y 394 Height 0

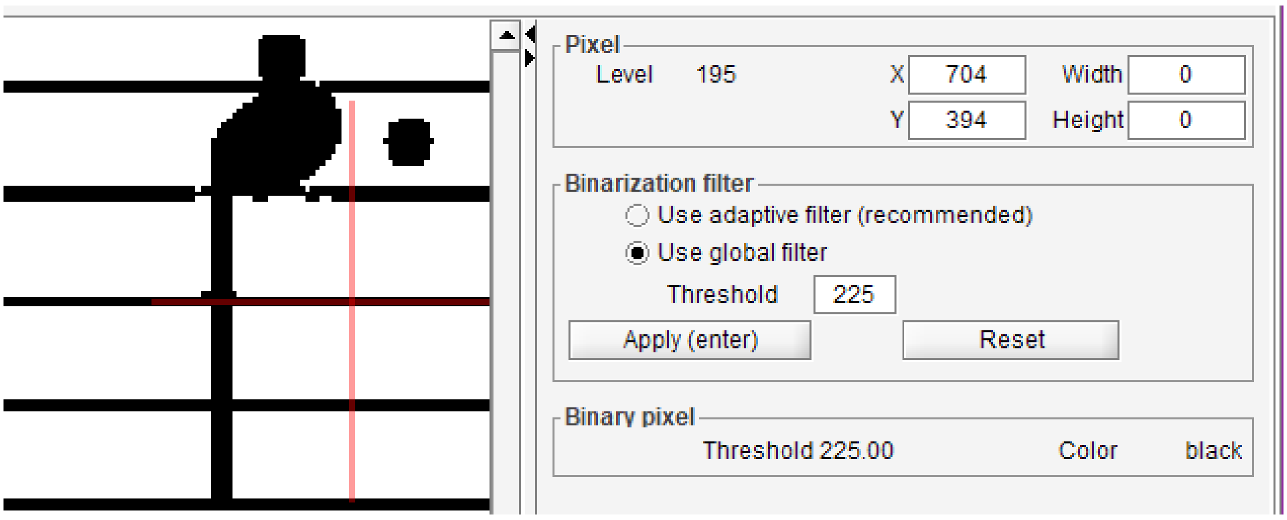
Binarization filter
○ Use adaptive filter (recommended)
● Use global filter
Threshold 140

Binary pixel
Threshold 140.00 Color white

Tuned global filter

Here, we have raised the threshold of the global filter from 140 to 225.

The binarized image is now OK for the next processing steps:



Pixel
Level 195 X 704 Width 0
Y 394 Height 0

Binarization filter
○ Use adaptive filter (recommended)
● Use global filter
Threshold 225

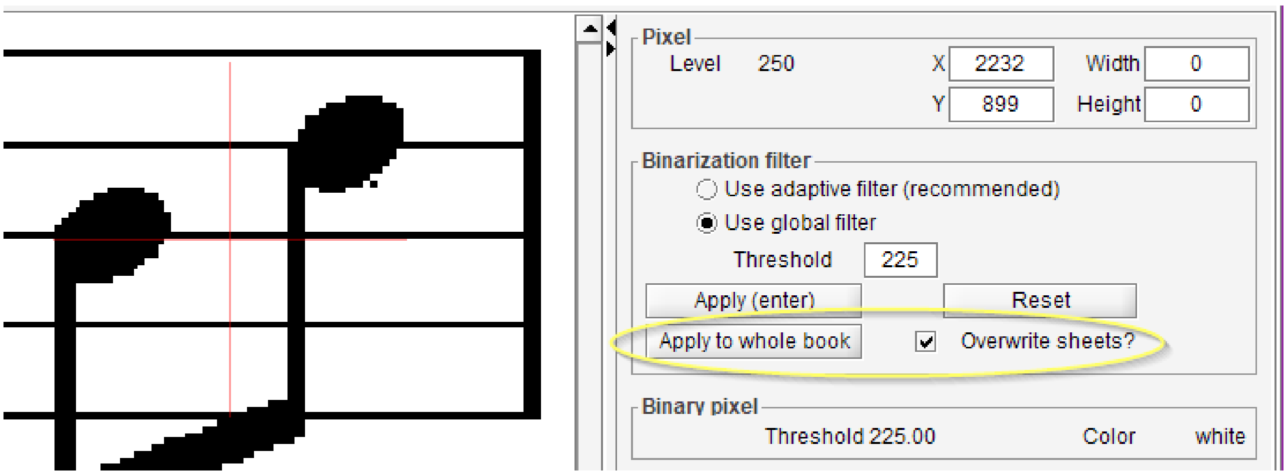
Binary pixel
Threshold 225.00 Color black

Case of a multi-sheet book

As we adjust the binarization filter and settings, we can observe the results on the current sheet.

If the sheet is part of a book containing several sheets, we have the ability to extend the binarization parameters to the entire book.

In such multi-sheet book, the binarization board displays an additional line, meant for the book level:



Pixel

Level 250 X 2232 Width 0

Y 899 Height 0

Binarization filter

○ Use adaptive filter (recommended)

● Use global filter

Threshold 225

☑ Overwrite sheets?

Binary pixel

Threshold 225.00 Color white

The button `Apply to whole book` extends the scope of the binarization parameters to the whole book.

- By default, this applies to all sheets is the book, **except** for the sheets which have specific binarization parameters.
- However, if the option `Overwrite sheets?` is ticked, this extension will remove any specific binarization parameter defined at a sheet level, thus making **all** sheets inherit from the same (book) parameters.

Binary board

This board, together with the Pixel board, is meant to visualize the results of the binarization algorithm documented in the BINARY step.

The board is displayed by default in the Gray tab and in the Binary tab.

- The *pixel* board displays the selected *location* (x, y). It can also display the *gray level* at this location when this information is available.
- The *binary pixel* board always displays the *threshold* value and the resulting *binary color*.

Threshold

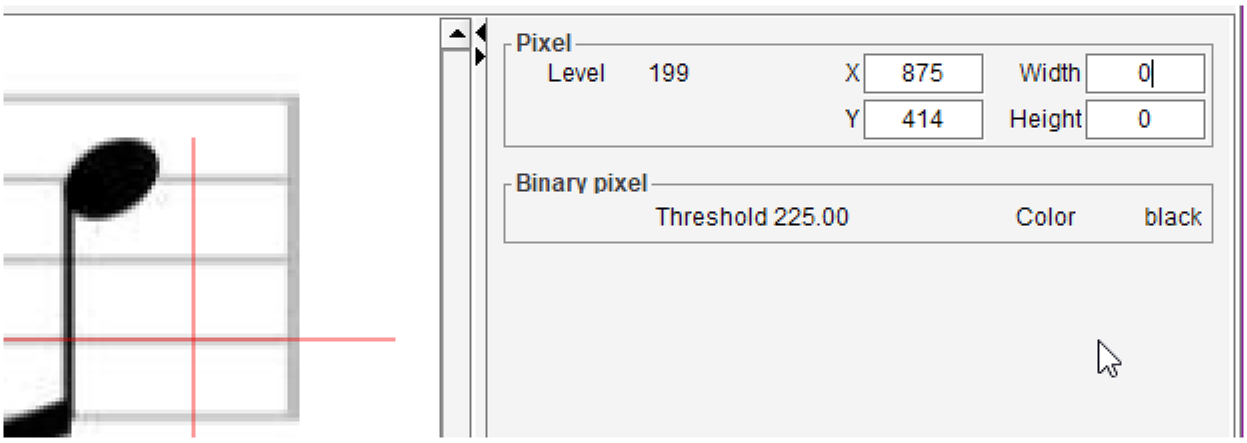
This field displays the threshold value to be compared with the gray level at the selected location.

Color

This field displays the resulting color of binarization at the selected location:

- *Black* if the pixel value is less than, or equal to, the threshold value,
- *White* otherwise.

Board for the Global filter



Pixel

Level 199 X 875 Width 0

Y 414 Height 0

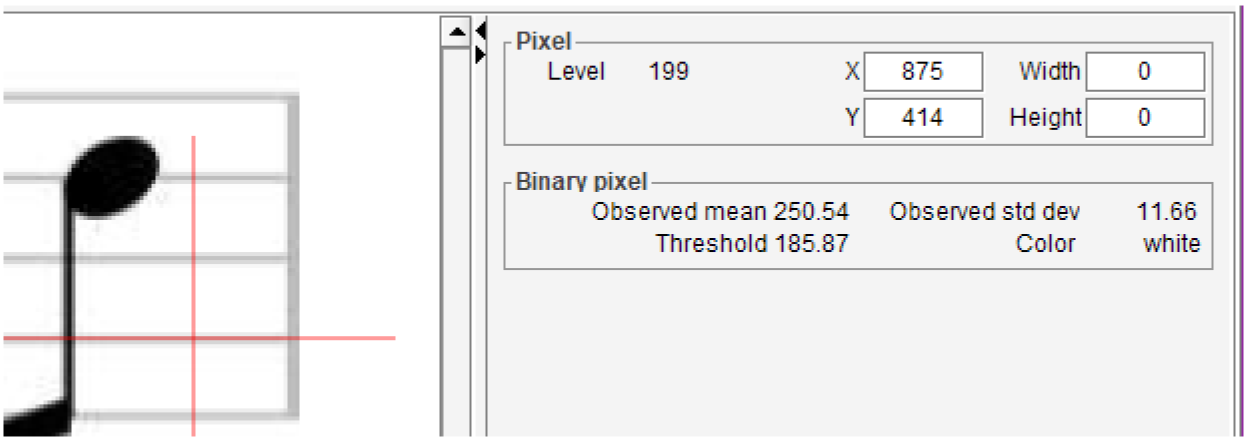
Binary pixel

Threshold 225.00 Color black

For the global filter, the threshold value is constant for all pixels in the image.

Nota: this example uses a global value (225) much higher than the default global value (140).

Board for the Adaptive filter



Pixel

Level	199	X	875	Width	0
		Y	414	Height	0

Binary pixel

Observed mean	250.54	Observed std dev	11.66
Threshold	185.87	Color	white

For the adaptive filter, the threshold value is computed for each location, based on the mean and the standard deviation of all gray values read around the selected location.

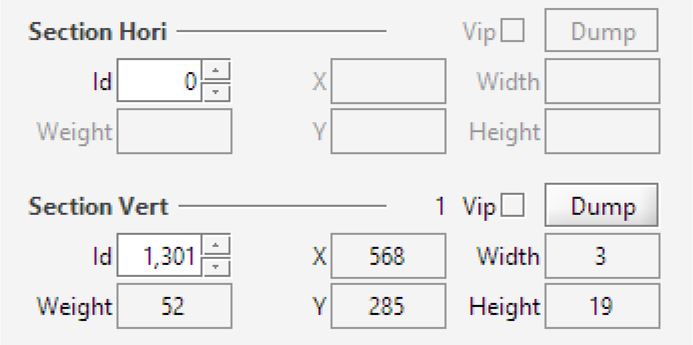
Observed mean

This field displays the average value observed in location vicinity.

Observed std dev

This field displays the standard deviation value observed in location vicinity.

Section board



Section Hori				Vip □	Dump
Id	0	X		Width	
Weight		Y		Height	
Section Vert			1	Vip □	Dump
Id	1,301	X	568	Width	3
Weight	52	Y	285	Height	19

There are two different section boards, one for sections with horizontal runs, one for sections with vertical runs.

TABLE OF CONTENTS

Vip

(input/output)
Flag this entity as VIP, resulting in verbose processing information.

Dump

(input)
Dump main entity data into the log window.

Id

(input/output)
Integer ID of entity.

Weight

(output)
Number of (black) pixels that compose the section.

X & Y

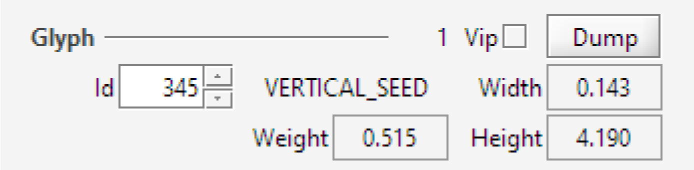
(output)
Coordinates of top left corner of the section:

- For a horizontal section, this is the left side of the top run.
- For a vertical section, this is the top side of the left run.

Width & Height

(output)
Width and height of the section bounding box.

Glyph board



Glyph 1 Vip □ Dump

Id 345 VERTICAL_SEED Width 0.143

Weight 0.515 Height 4.190

TABLE OF CONTENTS

Vip

(input/output)
Flag this entity as VIP, resulting in verbose processing information.

Dump

(input)
Dump main entity data into the log window.

Id

(input/output)
Integer ID of entity.

(groups)

(output)
If any, the group(s) this glyph is part of. A group is a tag assigned to a glyph, related to the intended usage of the glyph. For example, `VERTICAL_SEED` is assigned to a glyph considered for the detection of stem candidates.

Weight

(output)
The *normalized* glyph weight.

The glyph raw weight (number of black pixels it is composed of) is divided by the square of interline value to provide the interline-normalized weight.

Width & Height

(output)
The *normalized* dimension of the glyph bounding box (raw dimension divided by interline value).

`Inter` board

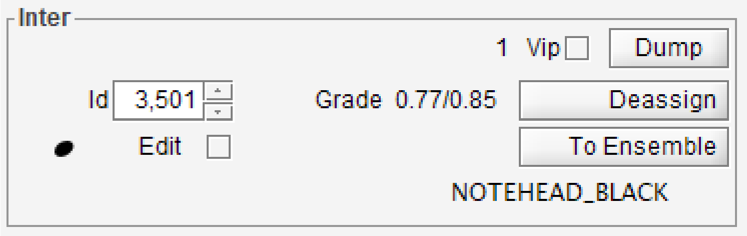


TABLE OF CONTENTS

Vip

(input/output)
Flag this entity as VIP, resulting in verbose processing information when entity is involved.

Dump

(input)
Dump main entity data into the log window.

Id

(input/output)
Integer ID of entity.

Grade

(output)
The intrinsic grade value assigned to the `Inter` instance. Followed by the computed contextual grade, if any.

Deassign

(input)
Button available to manually delete this interpretation.

(shape icon)

(output)
If available, the icon related to the `Inter` shape.

Edit

(input/output) Allows to open an editor of the entity.

To Ensemble

Button to navigate to the ensemble if any this entity is part of.

(shape name)

(output)
Name of the shape assigned to the `Inter` instance.

(word)

(input/output)

For a word inter, the modifiable word textual content.

(sentence)

(output)

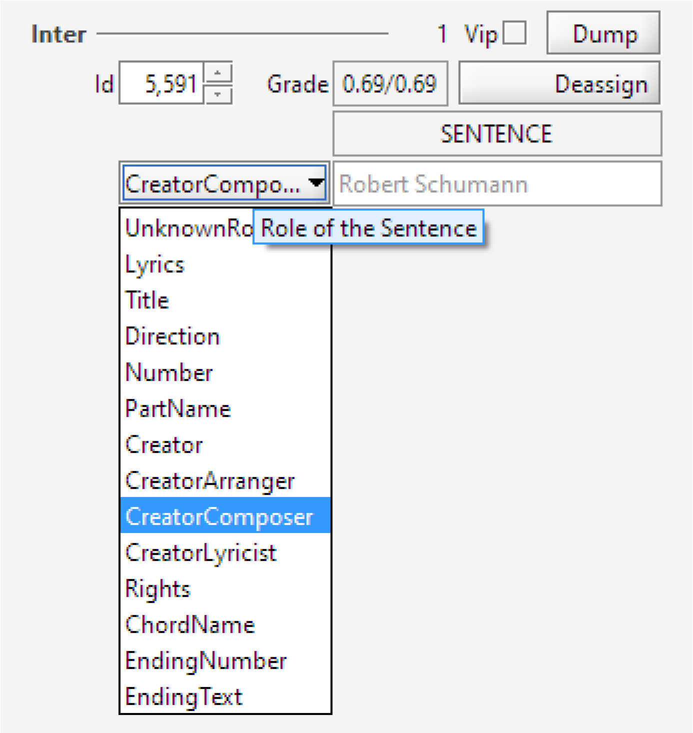
For a sentence inter, the *non-modifiable* sentence textual content.

Modifications can be performed at word level only.

(sentence role)

(input/output)

Role of the sentence (such as Direction, PartName, Rights, Lyrics, …).



Inter 1 Vip □

Id 5,591

Grade 0.69/0.69

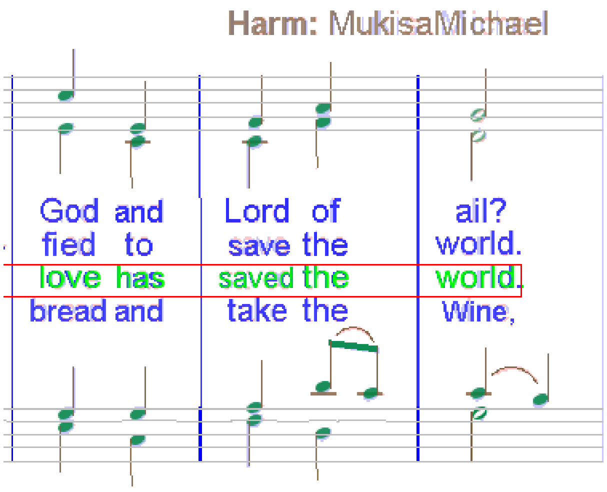
SENTENCE

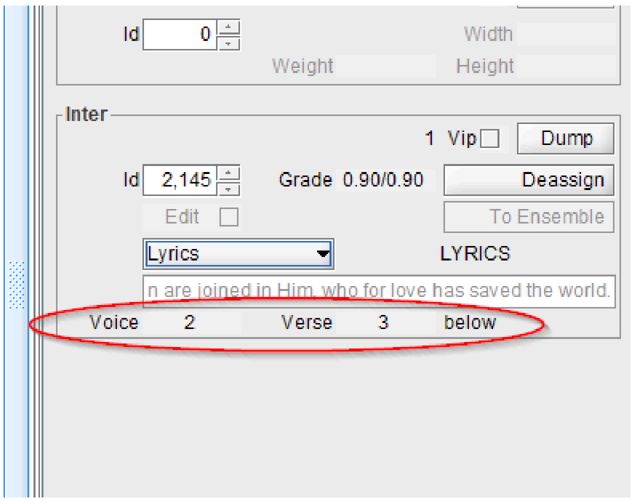
CreatorCompo…

Robert Schumann

Nota: `Lyrics` is such a specific sentence role that it cannot be changed in an existing `Inter`. Instead, a new (lyrics) inter is created automatically.

(lyrics)





Id 0 Width Weight Height

Inter

1 Vip □ Dump

Id 2,145 Grade 0.90/0.90 Deassign

Edit □ To Ensemble

Lyrics LYRICS

n are joined in Him, who for love has saved the world.

Voice 2 Verse 3 below

For a lyrics sentence, the inter board displays additional data:

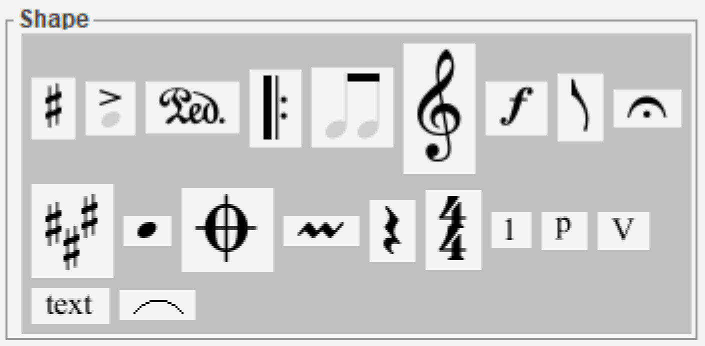
- Voice number,
- Verse number,
- Location with respect to staff.

Shape board

This board allows to handle about 200 different shapes.
To this end, the shapes are organized into some 20 shape-sets, and each shape-set is presented in a dedicated palette.

TABLE OF CONTENTS

Catalog of all shape-sets



The shape board initially displays the catalog of all shape-sets.

In the picture above, we can see:

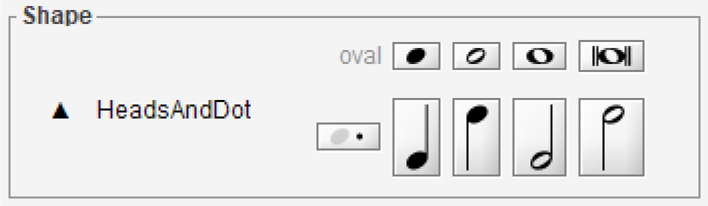
- Accidentals, Articulations, Attributes, Barlines, BeamsAndTuplets, Clefs, Dynamics, Flags, Holds,
- Keys, HeadsAndDot, Markers, Ornaments, Rests, Times, Digits, Pluckings, Romans,
- Texts, Physicals.

From this catalog, displayed with a dark background, no action like a drag n' drop can be launched. The purpose of the catalog is only to choose a shape-set.

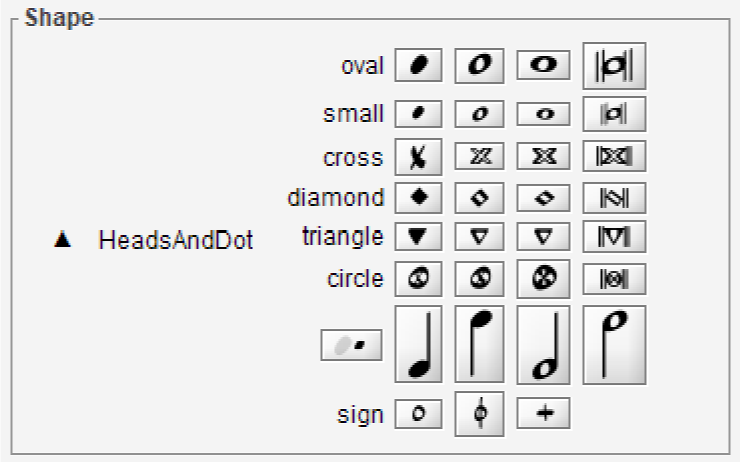
One shape-set palette

Clicking on a shape-set button replaces the global catalog by the selected shape-set, presented in a dedicated palette.
For example, clicking on the `HeadsAndDot` button will display the `HeadsAndDot` palette, whose content adapts to the book at hand:

Here is a simple configuration



And here is a more complex configuration for drums notation.
See the Drums chapter for further details.



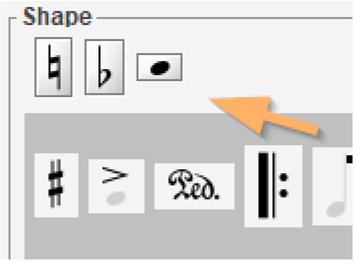
From any shape palette we can:

- Assign a shape to the current glyph, via a double-click on the proper shape button;
- Initiate a drag & drop action, by pressing the proper shape button and dragging it to the desired location in sheet.

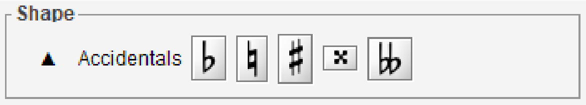
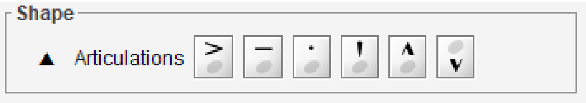
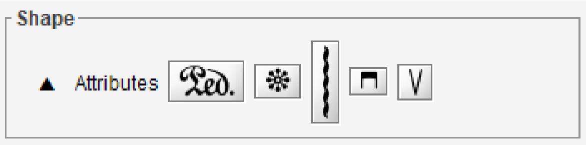
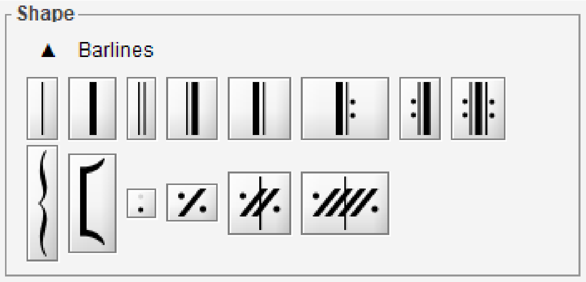
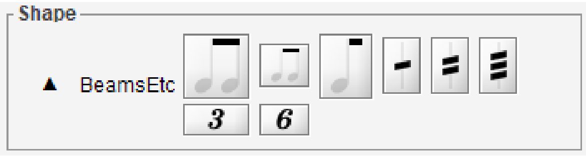
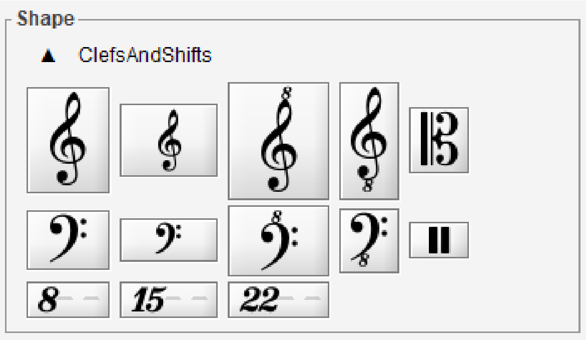
To leave the current palette and return to the global shape-set catalog, we press the `ESCAPE` key or click on the `up` (▲) button.

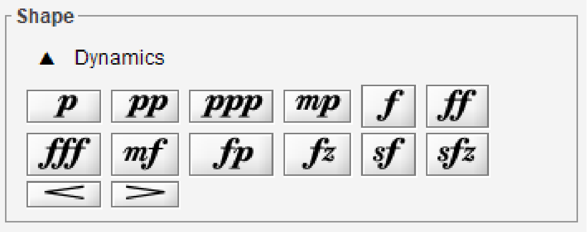
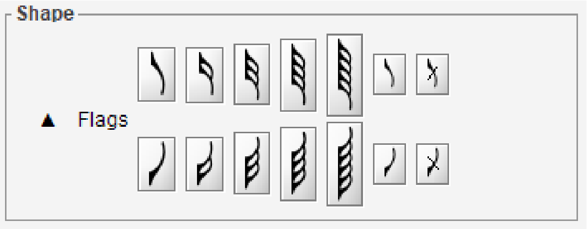
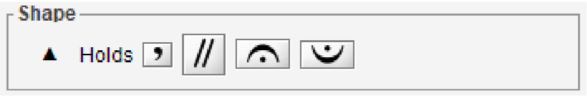
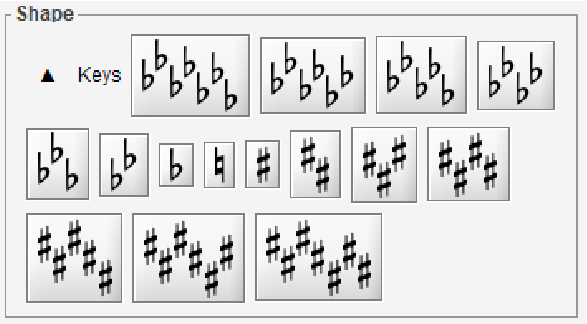
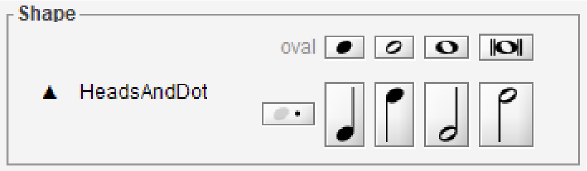
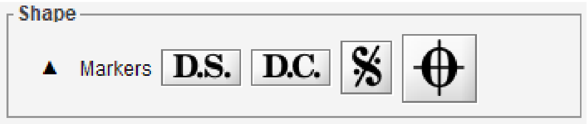
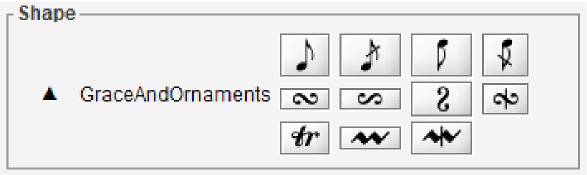
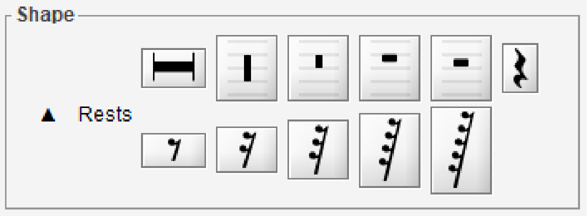
Recently used shapes

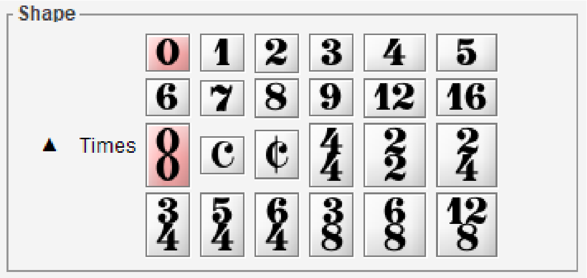
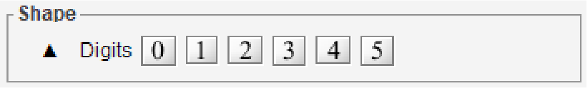
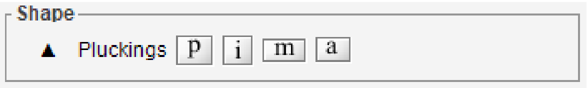
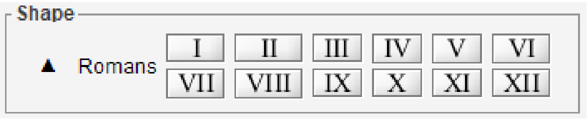
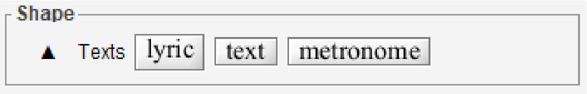
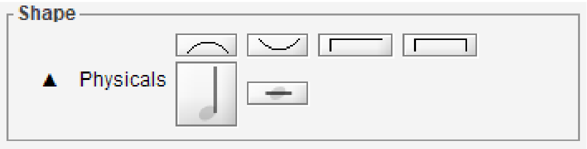
The shapes most recently used (by whatever means) always appear at the top of the shape board, making them easily available for a direct reuse.



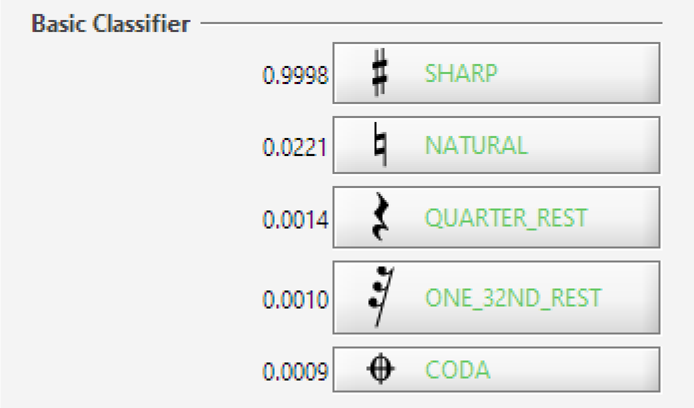
Palettes contents

Palette name	Palette content
Accidentals	
Articulations	
Attributes	
Barlines	
BeamsEtc	
ClefsAndShifts	

Palette name	Palette content
Dynamics	 Shape Dynamics p pp ppp mp f ff fff mf fp fz sf sfz
Flags	
Holds	
Keys	
HeadsAndDot	
Markers	
GraceAndOrnaments	
Rests	

Palette name	Palette content
Times	
Digits	
Pluckings	
Romans	
Texts	
Physicals	

Basic Classifier board



In current Audiveris version, the basic classifier is a Glyph classifier.

If a glyph is selected by whatever means, it is automatically submitted to the classifier and the top 5 shapes are displayed as active buttons, ordered by decreasing grade value.

Notice that the sum of grade values may exceed 1.0 (because no SoftMax filter is applied).

A simple click on a shape button assigns the related shape to the glyph at hand. (To be strictly correct, an `Inter` instance is created with proper shape and glyph).

Folders

This chapter describes the various Audiveris application folders on the target machine, depending on the operating system.

To allow its use by different users on the same machine, Audiveris always stores information in *user-specific* locations.

For such user-specific information, there are further distinctions between:

1. *Standard* information, such as the various score `data` output files, which are the only files any user can directly work upon.
2. *Essential* information, such as `config` or `train` files, which impact the application behavior and which should be modified only by an advanced user.
3. *Cached* (non-essential) information, such as `log`, `temp` or `gui` files, which are not meant to be edited.

These distinctions are enforced as much as possible, using the operating system features. For a general presentation, you can refer to the X Desktop Group and especially the XDG base directory specification.

TABLE OF CONTENTS

Standard folders CHANGED IN 5.4

This is where Audiveris stores all score outputs, such as the `.omr` project files, the `.pdf` printouts, the `.mxl` MusicXML files, etc.

Of course, we always have the ability to interactively store output by using the commands that prompt for a target location, like:

- `Book → Print book as`
- `Book → Export book as`
- `Book → Save book as`

Also, if we load an existing `.omr` file, perhaps to further modify it, it will be saved by default to the same location it was loaded from.

Now, if we are processing some input file, say `foo.pdf`, the question is: where will its output files (`foo.omr`, `foo-print.pdf`, `foo.mxl`, …) be stored *by default*?

The target locations are presented in the following sections, by decreasing priority.

TABLE OF CONTENTS

Command line option

The CLI option `-output <output-folder>`, if present, defines the target folder for every output.

For instance, the transcription of our `foo.pdf` input will be stored by default as

`<output-folder>/foo.mxl`

This specific CLI option overrides the general `Preferences` described in the sections below.

Preferences dialog

The Preferences dialog is accessible via `Tools → Preferences`.

Input sibling

The `Preferences` dialog offers an option named `Input sibling`.

When this option is set, any output is stored next to its input.

For instance, the transcription of `/some/path/to/foo.pdf` will be stored as `/some/path/to/foo.mxl` (assuming the folder `/some/path/to/` is writable).

Browse

If the `Input sibling` option is OFF, the `Browse` button allows to select an output folder.

For instance, if we have selected the output folder `/my/folder/`, the transcription of `/some/path/to/foo.pdf` will be stored as `/my/folder/foo.mxl`.

Separate folders

If the `Input sibling` option is OFF, the `Separate folders` option allows to gather all the outputs of an input file into a separate folder named according to the input radix.

For instance, still using the `/some/path/to/foo.pdf` input, the various outputs will be gathered into the specific folder `/my/folder/foo/`, thus resulting in something like:

```
my/folder/
├── foo/
|   ├── foo.omr
|   ├── foo.mxl
|   └── foo-samples.zip
├── ...
...
```

Essential folders

This is where Audiveris stores user-specific essential parameters:

You can create or modify these files, provided you are an advanced user and know what you are doing.

TABLE OF CONTENTS

Config folder

Audiveris defines a `CONFIG_FOLDER` for configuration files:

File name	Description
run.properties	User-modified application constants
logback.xml	Logging configuration
plugins.xml	Definition of plugins to external programs See the Plugins section.
user-actions.xml	Additional GUI actions

The precise location of `CONFIG_FOLDER` depends on OS environment:

OS	`CONFIG_FOLDER`
Windows	%APPDATA%\AudiverisLtd\audiveris\config
Linux (choice #1)	$XDG_CONFIG_HOME/AudiverisLtd/audiveris

OS	`CONFIG_FOLDER`
Linux (choice #2)	$HOME/.config/AudiverisLtd/audiveris
Flatpak	$HOME/.var/app/org.audiveris.audiveris/config
macOS	$HOME/Library/Application Support/AudiverisLtd/audiveris

Tessdata folder

Starting with 5.4 release, the language files needed by Tesseract OCR software, are hosted in the `tessdata` sub-folder of `CONFIG_FOLDER`.

To populate this `tessdata` folder, the interactive user can simply use the `Tools → Languages` pull-down menu.

Train folder

There is a `TRAIN_FOLDER` that can be populated with user-specific training material and trained model to override default Audiveris model:

File name	Description
basic-classifier.zip	Trained model and norms for the glyph classifier
samples.zip	Global repository of training samples
images.zip	Background sheet images of training samples

`TRAIN_FOLDER` is defined as the direct `train` sub-folder of `CONFIG_FOLDER`.

Cached folders

TABLE OF CONTENTS

Purpose

Audiveris uses these locations for persistency of internal information.

These files are **not** meant to be edited, period!

GUI folder

These are opaque files used for GUI lifecycle (notably last position and size of each window)

- AudiverisMainFrame.session.xml
- LanguagesFrame.session.xml
- LogicalPartsEditor.session.xml
- etc…

OS	`GUI folder`
Windows	%APPDATA%\AudiverisLtd\audiveris
Linux	$HOME/.audiveris
Flatpak	$HOME/.audiveris

OS	`GUI folder`
macOS	TODO: !!! **I DON'T KNOW** !!!

Log folder

Each session of Audiveris application creates in this folder a single global `dateTtime.log` file that covers all log events of the session.

OS	`LOG_FOLDER`
Windows	%APPDATA%\AudiverisLtd\audiveris\log
Linux (choice #1)	$XDG_CACHE_HOME/AudiverisLtd/audiveris/log
Linux (choice #2)	$HOME/.cache/AudiverisLtd/audiveris/log
Flatpak	$HOME/.var/app/org.audiveris.audiveris/cache/log
macOS	$HOME/Library/AudiverisLtd/audiveris/log

Temp folder

It is a temporary storage for various needs, such as saving a snapshot of a score image portion.

`TEMP_FOLDER` is defined as the direct `temp` sub-folder of `LOG_FOLDER`.

Outputs

Output files are generally located in the dedicated subfolder of the related book. See the Folders chapter.

This chapter presents the various output formats designated by their file extension.

A special emphasis is put on the `.omr` format, since this format governs the internal structuring of any Audiveris project file. Such file is also named a Book file, because there is one `.omr` project file per Book, and the file merely represents the marshalling of Book internal data from memory to disk.

Output Formats

There are 3 possibilities for the output of a transcribed score:

- Output as **OMR** file for saving and later reloading of OMR data for additional processing, manual correction or production of other outputs.
  See the .omr format section.

- Output as **PDF** file for direct use by a musician. It writes the resulting image into a PDF file, which is basically the content of the Picture/Binary tab in logical mode.
  See the .pdf format section.

- Output as **MusicXML** file for use in a score notation program. It writes a MusicXML file with the exported score entities.
  See the .mxl format section.

TABLE OF CONTENTS

- .pdf
- .zip

.log files

TABLE OF CONTENTS

Session global log

All messages displayed in Audiveris events window during the same session are also written into a separate log file located in Audiveris log folder (see Cached Folders).

The log file is named like `20180725T185854.log`, according to the date and time of session start, formatted as “yyyymmdd” + T + “hhmmss”.

Such log is a simple text file meant for later analysis. In particular, it is very useful when filing a bug report on Audiveris Issues site.

The advanced user can precisely customize the logged information by manually editing the logging configuration file `logback.xml` located in Audiveris config folder (see Essential Folders).

Batch books logs

When running in batch, Audiveris can process hundreds of books in a row. As mentioned above, this results in one global log file, perhaps a huge one.

So, in addition to the global log file, the batch processing of each book stores all the book-related log events in a separate book log file.

This book log file has a name formatted as `bookname-dateTtime.log`. It is located in the book subfolder itself.

This eases later analysis of batch processing for any specific book.

.mxl files

TABLE OF CONTENTS

Purpose

These are zip-compressed files of XML music files formatted according to **MusicXML** specification.

Most, if not all, music editors are able to import / export this kind of file, making MusicXML format a *de facto* standard for music information exchange. Please refer to the detailed MusicXML specification.

> Instead of `.mxl` files, which are compressed XML files, Audiveris can export plain (non-compressed) `.xml` files.
> To do this, we just set the option `org.audiveris.omr.sheet.BookManager.useCompression` to false.

Book level

A **book** export provides:

- Either one separate `.mxl` file for each movement found in the book (this is the default).
- Or a single `.mxl` file for a global `opus` containing the separate movements. The notion of opus is fully defined by MusicXML, but not supported by many editors, if any. To make Audiveris use opus, we simply set the option `org.audiveris.omr.sheet.BookManager.useOpus` to true.

Sheet level

A **sheet** export provides:

- One separate `.mxl` file for each page found in the sheet.

.omr files

A `.omr` file is the Audiveris *project* file for a given input file containing one or several images.

The project gathers all OMR information related to a given *Book* (one book for an input file), including its contained *Sheet* instances (one sheet for each image).

The project file is merely the result of the XML marshalling of a book and its sheets, so there is no logical difference in the project structure in memory and on disk. In memory there are a few additional transient variables, meant only for processing speedup.

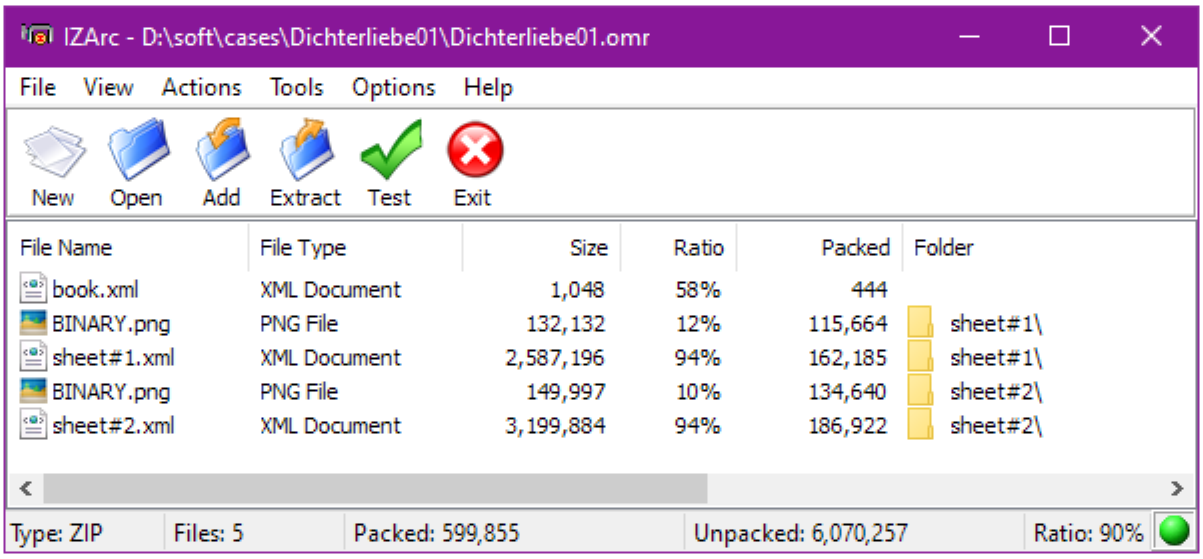
This identical structuring between memory and disk allows the gradual processing of books of any size, since data can be continuously saved to (and restored from) disk while sheets are being processed.

All other Audiveris outputs derive from this project data.

File structure

The internal project structure is specific to Audiveris software but it is not opaque and is meant for public access, either via the Audiveris API or by the direct extraction of XML fragments.

The file is a collection of XML files gathered into a **Zip** archive, whose content can be easily browsed (and even modified – *at your own risk!…*) via any Zip reader or editor.



IZArc - D:\soft\cases\Dichterliebe01\Dichterliebe01.omr

File Name	File Type	Size	Ratio	Packed	Folder
book.xml	XML Document	1,048	58%	444	
BINARY.png	PNG File	132,132	12%	115,664	sheet#1\
sheet#1.xml	XML Document	2,587,196	94%	162,185	sheet#1\
BINARY.png	PNG File	149,997	10%	134,640	sheet#2\
sheet#2.xml	XML Document	3,199,884	94%	186,922	sheet#2\

Type: ZIP | Files: 5 | Packed: 599,855 | Unpacked: 6,070,257 | Ratio: 90%

The picture above is a Zip view on a 2-sheet book (Dichterliebe01.omr example).
Here below is the same example, presented as a tree:

```
├── book.xml
├── sheet#1
│   ├── BINARY.png
│   └── sheet#1.xml
└── sheet#2
    ├── BINARY.png
    └── sheet#2.xml
```

As you can see, we have in this `.omr` file example:

- A single `book.xml` file
- One `sheet#N` subfolder for each sheet in the book, which here contains:
    - A `BINARY.png` file
    - A `sheet#N.xml` file

Beyond this small example, here below are all the possible file kinds:

File Name	Content
`book.xml`	Skeleton of book hierarchy
sheet#N/	Specific folder for sheet #N
sheet#N/`BINARY.png`	Black and white image of sheet #N
sheet#N/`GRAY.png`	Gray image of sheet #N
sheet#N/`HEAD_SPOTS.png`	Head spots image for sheet #N

File Name	Content
sheet#N/sheet#N.xml	OMR details for sheet #N

You may have noticed two files that did not appear in our small example, namely GRAY.png and HEAD_SPOTS.png:

- GRAY.png image is built from the sheet original input image (which may be a colored image, perhaps with an alpha channel) and results in pixel gray values.
  The BINARY step processes this gray image to produce the black and white BINARY.png image, which is needed by all subsequent steps. By default, the gray image is then discarded.
  One case where the GRAY.png image would still be useful is when the user would want to modify the image geometry, typically by rotating or de-warping – features not yet provided as of this writing. In that case, the quality would be better preserved when operating on gray rather than binary image.

- HEAD_SPOTS.png image is a temporary information carried over from the BEAMS step to the HEADS step. It takes advantage of the image processing performed during the BEAMS step to provide the image areas (spots) where black heads are likely to be recognized.
  This image is discarded at the end of the HEADS step.

Memory constraint

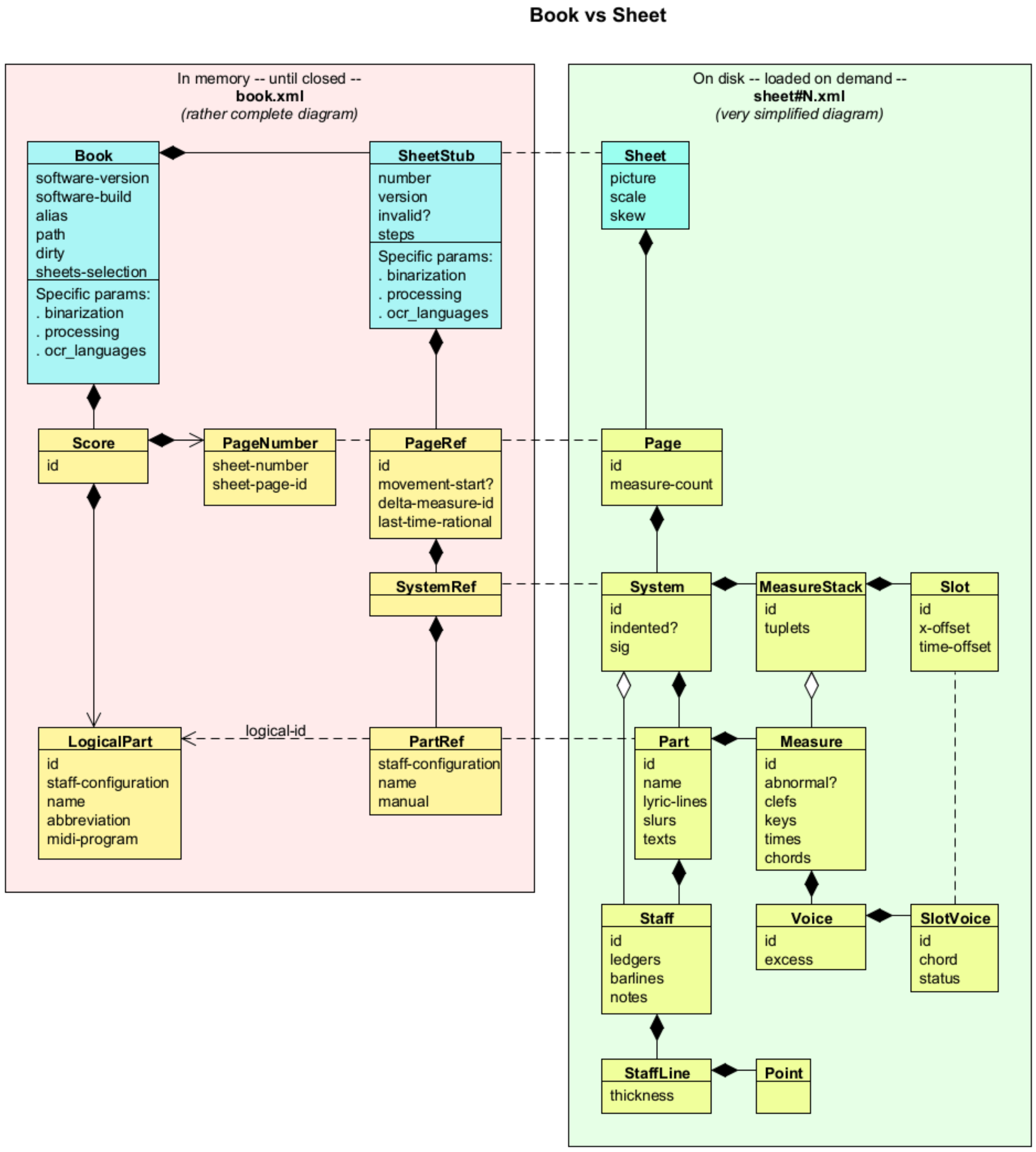
To be able to handle books with more than a few sheets, Audiveris keeps book top data in memory, while sheet “details” are loaded only when they are really needed.

In terms of project file, this means that the book.xml part is kept in memory until the book is closed, while the various sheet#N/BINARY.png and sheet#N/sheet#N.xml are loaded on demand.

So, the physical and logical containments are implemented as follows:

- A Book does not directly contain Sheet instances but only a sequence of **SheetStub** instances (or Stub for short) with a one-to-one relationship between SheetStub and Sheet.
- In a similar manner, a Score actually contains **PageRef** instances which refer to Page instances.

In the following class diagram, physical entities are displayed in blue, and logical entities in yellow.



On the left side, the `book.xml` artifact represents the content of the *book.xml* part within an Audiveris project file, and is kept in memory. Note that the `Score` and `PageRef` instances can exist only when the `GRID` step has been reached on (some sheets of) the book.

On the right side, the `sheet#N.xml` artifact represents one sheet (perhaps among others), pointed to from *book.xml*, and is not necessarily in memory.

> **NOTE**
>
> Just to avoid endless clashes with the Java predefined `System` class, we have chosen `SystemInfo` as the class name to represent a music *system*.

Schemas documentation

We wanted to make `.omr` file as open as possible to let end-users and developers use directly this file for any purpose compliant with the Audiveris AGPL license.

To this end, we have developed a specific documentation set.
For each of the two kinds of XML files (`book.xml` and `sheet#N.xml`) we provide:

1. The corresponding formal schema description as a `.xsd` file.
   This `.xsd` file can be further used to automatically:
   - Generate a binding between the XML file and a coding language such as Java.
   - Validate the XML file produced by another source such as a test program.
2. The corresponding schema documentation as a `.html` file meant for a human reader. It documents the various elements and attributes of the XML hierarchy, augmented by JavaDoc annotations from Audiveris source code.

Since Audiveris 5.3, the Split & Merge feature can accept play-list definitions created via Audiveris UI or provided as XML files created by any plain text editor.
Consequently, we provide an additional `.xsd` + `.html` pair, this time for a `playlist.xml` file.

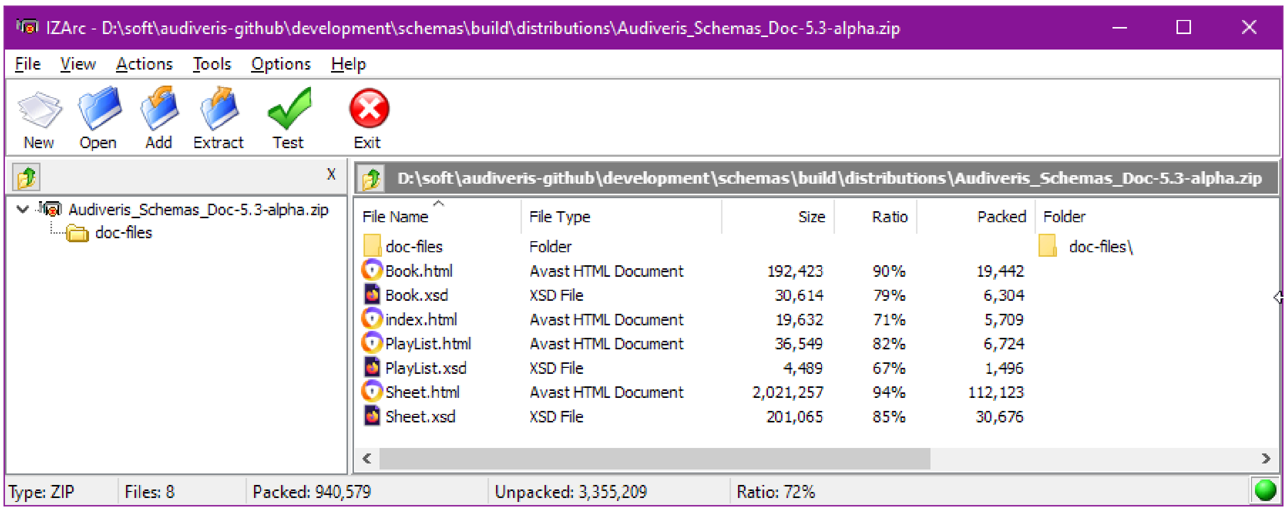
Documentation distribution

We could not integrate the schemas documentation set into Audiveris HandBook on GitHub Pages.
Please tell us if we are wrong, but we could not use GitHub Pages for two reasons:

- Putting large `.html` files under GitHub Pages source control would be an awful waste of resources
- GitHub Pages would serve these `.html` files with incorrect mime type.
  They would be served as plain text files, preventing suitable rendering by any web browser.

So, for the time being, the Audiveris schemas documentation is packed as a ZIP archive and simply published side by side with the most recent releases available on the Audiveris Releases section at GitHub site. The typical file name is `Audiveris_Schemas_Doc-X.Y.Z.zip`, where X.Y.Z is the precise release version.

For example, here is the content of a recent version:


IZArc - D:\soft\audiveris-github\development\schemas\build\distributions\Audiveris_Schemas_Doc-5.3-alpha.zip
File View Actions Tools Options Help
New
Open
Add
Extract
Test
Exit
Audiveris_Schemas_Doc-5.3-alpha.zip
doc-files
D:\soft\audiveris-github\development\schemas\build\distributions\Audiveris_Schemas_Doc-5.3-alpha.zip

File Name	File Type	Size	Ratio	Packed	Folder
doc-files	Folder				doc-files\
Book.html	Avast HTML Document	192,423	90%	19,442	
Book.xsd	XSD File	30,614	79%	6,304	
index.html	Avast HTML Document	19,632	71%	5,709	
PlayList.html	Avast HTML Document	36,549	82%	6,724	
PlayList.xsd	XSD File	4,489	67%	1,496	
Sheet.html	Avast HTML Document	2,021,257	94%	112,123	
Sheet.xsd	XSD File	201,065	85%	30,676	

Type: ZIP | Files: 8 | Packed: 940,579 | Unpacked: 3,355,209 | Ratio: 72%

You can simply download and expand this ZIP archive locally, to get an easy access to its content.

- The `.xsd` files are available for a picking.
- The HTML documentation can be browsed from the `index.html` root file. The various HTML pieces use links to images stored in the `doc-files` sub-folder.

Documentation generation

If, as a developer, you want to (re-)generate this documentation set on your own, please refer to the Building Schemas Documentation article in Audiveris Wiki.

.pdf files

Audiveris can print out a whole book or a single sheet.

In both cases, it provides a `.pdf` file, located by default in the book subfolder.

The pdf file contains one page per sheet and its content is basically the logical output of the Data tab, with no annotations and no colored voices.

If no recognition error remains, this pdf should very closely look like the original image.

.zip files

TABLE OF CONTENTS

Purpose

These files are containers mainly used for training material.

Book

A book folder (let’s name it “BOOK”) may contain:

- **BOOK-samples.zip**: This is the book samples repository. It gathers all samples (glyph + shape) collected in the book. The advanced user should carefully review (and perhaps edit) all the samples of this *local* repository, before pushing its content to the *global* repository.
- **BOOK-images.zip**: This is an optional companion of the local samples repository. It contains the related image of each sheet for which at least a sample is present. This allows to display the background context of the sample.
- **BOOK-annotations.zip**: This is just an output of Audiveris 5.x, meant for the future page and patch classifiers scheduled for 6.x versions. It provides sheet image and symbol annotations (shape + bounding box) for each recognized inter.

Global

These files are located in the `TRAIN_FOLDER`. See section on Essential folders.

samples.zip: This is the global samples repository. Its content is based on all local repositories that have been pushed to it. It is the only source for training the glyph classifier.

images.zip: This archive contains the images pushed with samples from local repositories. It can get fairly big, and is used only for visual check, not for training.

Limitations

This section presents the known cases that current Audiveris does not handle properly. These are “known limitations”, which should be addressed in future releases.

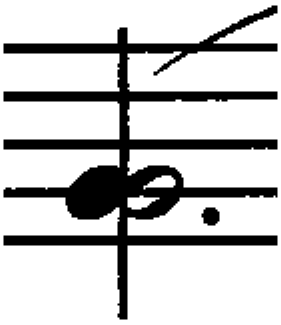
It is important that the end user be aware of these cases to avoid wasting time on them with the current release. Apart from these cases, you can file a bug report or a request for enhancement on https://github.com/Audiveris/audiveris/issues.

> Nota: There used to be a section on “*Natural signs in key signature*” and a section on “*Key signature change*” to report limitations on these topics. Starting with the 5.8 release, these limitations no longer exist. See details on the Key Signature chapter.

TABLE OF CONTENTS

Opposed Stems

We can have two head chords with up and down stems that are located in such a way that they seem to be merged, as follows:



The OMR engine may detect just one long stem, instead of two aligned ones. The problem is that this single stem has some heads attached near the middle of the stem, but no head on any stem end. At the next reduction, this stem will be discarded, and the now isolated heads as well.

This error can be fixed by manually inserting separate standard stems (and the related heads).

ChordName

Recognition of chord names can still be impeded by the presence of sharp (`♯`) or flat (`♭`) characters which Tesseract OCR cannot handle correctly.

As a workaround, we can manually replace them by number (`#`) and lowercase `b` characters. But the real solution should come from the training of Tesseract OCR on these embedded alteration signs.

See further description in the Text Chord Name section.

Tuplets

Tuplets other than triplets and 6-tuplets are not supported, even manually.

Roman numeral

Some input files represent chords using Roman numerals.

These are recognized, but not yet exported to MusicXML.

Reference / Updates

Software updates

As detailed in the Git Workflow article on Audiveris Wiki:

- Audiveris development is performed on the specific GitHub “**development**” branch.
  It is thus continuously available for anyone that pulls and builds the software from source.
- The default “**master**” branch is updated only for software releases.
  At this time only, a new release is created and uploaded to the Releases section on GitHub site.

TABLE OF CONTENTS

5.9

(December 2025)

- User Interface
  - The Preferences dialog let the user select new processing options.
- Engine
  - Ability to automatically swap a sheet from memory to disk once it has been processed.
  - Ability to process all systems of a sheet in parallel.
  - Support for Cyrillic OCR’d characters.
  - Better handling of staves that exhibit non-regular vertical spacing between lines.
  - Support for the Staccatissimo below shape.
- Project
  - Publication of Audiveris on the Windows Community repository via WinGet command.

5.8

(November 2025)

- User Interface
  - Improved key editor to build key signature from its members
  - Improved staff editor that inserts intermediate defining points where needed by staff extension
  - Improved ending editor that can resize the ending legs.
- Engine
  - Support for key signature changes
  - Support for strings techniques of up bow and down bow
  - Multi-measure rests can survive the lack of a recognized measure count
- Project
  - Use of ‘macos-15-intel’ instead of ‘macos-13’ runner to build the installer for macOS Intel
  - Flatpak to catch up with OS installers
  - Revamped the book/sheet annotations meant for training an external model
- Java
  - Upgrade to Java 25 and Gradle 9.1

5.7

(September 2025)

- User Interface
  - Support for CJK (Chinese, Japanese, Korean) characters
  - The Preferences and Constants dialogs now scale correctly with font size changing
  - Ability to print out the detailed steps durations for a sheet processing
- Engine
  - Correct handling of void regions in small note head templates
  - Improved computation of voices distances, resulting in better voices assignment
- Project
  - Additional Windows installer provided with console, useful for launch messages
  - All installers now come with a max heap set at 8GB
- Documentation
  - The developer-oriented schema documentation is now included as an asset in each release
- Java
  - Upgrade to Java 24 and Gradle 9.0

5.6

(May 2025)

- User Interface
  - Logical parts editor: manual selection of MIDI program for any logical part
  - Inter board: editing of word attributes
- Engine
  - Use of OCR’d word attributes to determine the text font family and style for any text word
  - Abandon of the manual selection of text font at global/book/sheet levels
  - Addition of Primus font into the collection of music fonts

5.5

(Apr 2025)

- Project
  - Use of jpackage/jlink utilities to build a JRE-included installer. Contributed by Fabio Marques.
  - Use of GitHub workflows to generate installers for Windows, Linux and macOS.
  - Abandon of the old JRE-free Windows installer
  - Generalized use of GitHub workflows to publish Audiveris releases assets (installers, sources, docs)

5.4

(Feb 2025)

- User Interface
  - New feature to handle the local collection of OCR languages and download additional ones on demand from the GitHub Tesseract site.
  - Creation of the Binarization adjustment board to dynamically adjust the binarization filters and settings. Contributed by Michael Porter.
  - Past the SYMBOLS step, the manual removal of any `Inter` now triggers a dynamic rebuilding of glyphs, as if the removed `Inter` had never existed.
  - Improvement and extension of default/book/sheet parameters (interline, barline)
  - Ability to display some inters in jumbo mode to ease a visual inspection – by default this applies to augmentation dots.
  - Ability to gracefully stop the current book processing at the next step end.
  - Ability to clear the log window – the log file remains intact.
  - Waiting message for long loading of book or sheet.
  - New `Preferences` dialog with new policy for the selection of output folder.
- Engine
  - Improved handling of implicit tuplets.
  - Support for metronome mark.
  - The recognition of fermata no longer requires separate recognitions of fermata-arc and fermata-dot.
  - The processing of an input image with no white margin around the

  staves is now possible.
  - Staff lines are now less impacted by isolated chunks.
- Project
  - A Linux Flatpak installer, gathering the needed libraries including proper Java environment, is provided on FlatHub. Contributed by Martin Wilck and Jan-Willem Harmannij.
  - The Windows installer pre-populates the user `config`/`tessdata` folder with the `eng` Tesseract language.
  - For licence reasons, JPodRenderer had to be replaced by Apache PDFBox to load PDF images.
- Documentation
  - Global handbook restructuring.
  - Support for a PDF version of handbook.
  - Re-reading of the whole documentation. Contributed by Brian Boe, native American-English speaker!
- Java
  - In `Audiveris.bat` (used by the Windows installer) and `Audiveris` start scripts, the Java version is checked before the application is launched.
  - Support of Java 21.
  - Upgrade to Gradle 8.5 to support Java 21.
  - Removal of all deprecated features such as JGoodies PanelBuilder, Observer/Observable, class.newInstance(), etc.

5.3

(Jun 28, 2023)

- User Interface
  - Editing of staff geometry, at individual line and global staff levels
  - User management of score logical parts, and their mapping to sheet physical parts
  - User editing of newly supported features (multi-measure rests, octave shifts, etc)
- Engine
  - Support for drums unpitched notation. Contributed by Brian Boe.
  - Support of several musical font families for better template matching
  - Support of several text font families
  - Support for multi-measure rests

  - Support for measure repeats for 1, 2 and 4 measures
  - Support for octave shifts (single and multi-line shifts)
  - Support for endings with no left leg, with default number generation
  - Support for two populations of beam height and head size
  - Support for beam thickness specification
  - Support for fingering and plucking
- Project
  - Use of Tesseract 5.x in legacy mode
  - Support of MusicXML 4.0
  - Generation of Schemas documentation
- Java
  - Support of Java 17

5.2

(Jan 18, 2022)

- User Interface
  - Ability to move and resize symbols
  - Snapping note heads to staff line/ledgers and stems
  - Repetitive input
  - Ability to merge/split books
  - Ability to merge/split parts via brace handling
  - Ability to merge systems
  - Ability to merge/split chords
  - Ability to modify voice or time slot for chords
  - Support for compound notes (head + stem)
  - Support for key signature cancel (key made of natural signs)
  - Improved support for chord names
  - Chords popup menu offers “Next In Voice” relation to better guide voice/slot assignments
  - Better handling of sentences, chord names, lyrics
  - Ability to correct beam thickness scale
  - Ability to limit book processing on selected sheets
  - Support for high DPI devices
  - Internationalization of all menus
- Engine

  - Better handling of poor-quality scores
  - Support for 1-line percussion staves
  - Detection of 4-line and 6-line tablatures
  - New algorithm for time slots and voices
  - Support for rest chords interleaved within beam head chords
  - Support for non measure-long whole rests
  - Support for implicit tuplets
  - Support for merged grand staff configuration
  - Support for cross heads
  - Improved tie slurs detection on staff, across systems and across pages
  - Plugin mechanism upgraded for multi-movement exports
- Project
  - Automated checks for update on GitHub
  - Documentation handled on GitHub Pages
  - How to use Gimp to improve input. Contributed by Baruch Hoffart.
  - How to enlarge low resolution scans. Contributed by Ram Kromberg.
- Java
  - Support of Java 11
  - Refined dependencies on Java EE, JAXB, etc away from Java 8

5.1

(Dec 13, 2018)

- User Interface
  - Visual separation of shared heads
  - User assignment of fixed-shape symbols
  - Ability to modify scale parameters
- Engine
  - Augmentation dots apply to shared heads
- Project
  - Windows binary installers (32 and 64). Contributed by Baruch Hoffart.
- Java
  - Support of Java 8

5.0

- Engine
  - Creation of `.omr` project files
- Project
  - Migration to GitHub & Gradle. Contributed by Maxim Poliakovski.

4.2

- Distribution Changes:
  - Installation: Major OSes Windows and Linux and machine architectures x86 and x64 are supported via dedicated installers. The installers take care of all Audiveris dependencies (Java runtime, C++ runtime, musical font, etc).
  - Nota: Support for macOS has been dropped from the scope of this release to avoid further delays.
  - OCR: A few selected languages are pre-installed with Audiveris distribution (deu, eng, fra, ita). Additional languages can be supported by downloading the related trained data from the dedicated Tesseract web page.
  - NetBeans: A pre-populated nbproject folder provides NetBeans support out-of-the-box.
- New Features:
  - OCR: Tesseract V3.02 has been integrated in place of oldish V2.04 version. This much more powerful engine has led to a global redesign of text handling within Audiveris. There is now a dedicated TEXTS step which performs a layout analysis on each whole system image and transcribes the identified textual items. Note also that several languages can be selected at the same time.
  - Binarization: Extracting foreground pixels from the background has long been performed using a global threshold on pixel gray value. Even images with non-uniform illumination can now be processed with an adaptive filter which takes into account the neighborhood of the current pixel.
  - Glyph recognition: The major part of neural network input consists in moments which capture glyph key characteristics. Former Hu geometric moments have been replaced by ART moments (Angular Radial Transform, as used by MPEG-7) which are less sensible to noise.
  - Plugins: Audiveris MusicXML output can be “piped” to external softwares

such as score editors or MIDI sequencers, through a flexible plugin mechanism. Consequently, these features have been removed from Audiveris application.

- Bug Fixes:
  - PDF input: Several free Java libraries have been tested (PDFRenderer for a long time, then JPedal and PDFBox) but none was really satisfactory. Hence support for PDF input is now delegated to a Ghostscript sub-process, a fully functional and perennial solution.
- Other Changes:
  - Doc: A comprehensive Handbook is now available from Audiveris web page, as well as the API JavaDoc of the current release. The former installation tab is now merged with the first chapter of the handbook.
  - Wiki: The online Audiveris Wiki contains detailed documentation about how to process each score of the set of examples available on MakeMusic/Recordare site. It is also used to gather information about evolutions being considered for Audiveris software.

4.1

- Distribution Changes:
  - Several installation files have been published, all using the 4.1beta core name. This reflects the status of continuous development rather than stable release of the software.
- New Features:
  - Filaments: They are long glyphs representing the core of either horizontal or vertical lines (staff lines candidates and barlines candidates respectively). These filaments are formalized in natural splines, which are sequences of Bézier curves with continuity up to the second derivative.
  - Grid: The staff lines and barlines are connected into a grid of sometimes rather wavy lines. The grid itself is taken as the referential for all the other glyphs, whatever the potential skew or other distortion of the image, and thus saving the need for any pre-processing. Moreover, one can on demand easily build and save a “dewarped” version of the initial image.
  - Scale: Additional key informations are derived from run length histograms (jitter on staff line thickness and spacing, typical beam height, whether the image is music or not).
  - Systems: The boundary between two consecutive systems is now a

  broken line, resulting from the incremental inclusion of glyphs into their nearest system.
  - Training: Besides full sheets taken as training samples, the user can select a mode that takes every manual assignment as a new training sample.
  - Symbols: The HEAD_AND_FLAG family of compound symbols no longer exists, thanks to an aggressive strategy in glyph split pattern.
- Known Issues:
  - OCR: We are still stuck to the old Tesseract version (2.04). The new Tesseract generation (3.x) has been out for more than one year now but still lacks a Java connection under Windows.
- Other Changes:
  - Time: All time values, such as offsets within a measure, are computed using rational values, which makes them independent of the score divisions value.

4.0

- New Features:
  - Display: The main application window has been simplified. Only two views are now shown for each sheet: Picture (focused on input image) and Data (focused on items detected). We no longer have separate windows for sheet and score. The score elements are displayed in a translucid manner on top of the sheet glyphs they represent, in order to visually catch any discrepency. Separate voices can be displayed each in a specific color
  - Every other window (Log, Errors, Boards) can be displayed or hidden, and each individual board can be selected at will
  - Font: Former symbol bitmaps have been dropped for the use of a TrueType music font (Stoccata.ttf then MusicalSymbols.ttf). This allows endless zooming of displays and printouts with no loss of quality
  - The font is even used to build artificial symbols used for initial training of the neural network
  - Print: Ability to print the resulting score into a PDF file
  - Multi-page: Multi-page images (using PDF or TIFF format) can be transcribed to multi-page scores in memory
  - A disk-based prototype version, using a map/reduce approach, allows to combine existing MusicXML pages into a single score

3.4

(Dec 14, 2012)

- Distribution Changes:
  - Libraries: All the external jars (23 as of this writing) needed to rebuild and/or run Audiveris are now provided in a dedicated /lib folder available in the download area. A developer can still pick up a newer jar version from the Internet.
  - Player: The XenoPlay MusicXML player has been replaced by a better player, named Zong!
- New Features:
  - Bench data: To allow the analysis of multiple batch runs, and compare the recognition efficiency, each sheet processing can log key data in a dedicated file. For the same purpose, time-out values can be defined for script or step processing.
  - Bar Lines: The user can now interactively assign / deassign a bar line that defines parts, thus recreating the systems from scratch.
  - Constants: All application constants can now be set from the CLI with the -option keyword. This complements the ability to set them from the Tools - Options UI menu.
  - Dots: Support for double dots, ability to assign the role of any dot (augmentation, repeat bar line, staccato)
  - Horizontals: Horizontal entities (ledgers, endings) can now be forced or deassigned manually.
  - MIDI Player: The MIDI playback is now driven from a separate console window, borrowed from Zong! player.
  - OCR: Tesseract OCR is now available under both Windows and Linux.
  - Score: From a dedicated Shape palette, the user can Drag n’ Drop a (virtual) glyph to either the score view or the sheet view, thus injecting entities directly into the score structure.
  - ScoreView: The zoom of the score view can now be adjusted at will, thanks to a slider and better symbol bitmap definitions. A next version will replace them with the use of Stocatta true-type font.
  - Time Signature: The user can now enter any custom time signature, defining numerator and denominator values explicitly.
  - Time Slots: Within a measure, the time slots are meant to gather notes that begin at the same moment in time. The user can now choose at the score level the policy for determining the time slots, either through stem alignment or through note head alignment.

  - Tuplets: 6-tuplets are now supported, as well as tuplets mixing beamed notes with other notes (flagged notes, rests, …).
  - UI: A new board (Shape palette) is available. It allows drag n’ drop for entity injection, easier navigation through shape ranges, and shape assignment by double-click.
  - UI: All boards now have an expand / collapse mechanism, thus allowing to save room in the column of boards.
- Bug Fixes:
  - Player: The Zong! Player is now more tolerant with respect to measure defects. It no longer throws an exception whenever the notes durations within a measure are not consistent with the measure expected duration.
  - Player Data: The data part of Zong player is now provided as a resource in a dedicated jar file, thus allowing the launching of Audiveris from any location of your computer.
  - Exception handler have been removed from all unitary tests, so that the results are clearly seen as successes or failures
- Known Issues:
  - Virtual Glyphs: For the time being, the (virtual) glyphs created by direct injection cannot be moved or resized once they have been dropped from the Shape palette to their target view. However, they can be deleted and re-injected (this workaround addresses a move but not a resize).
- Other Changes:
  - Images: Support for most pixel sizes.
  - Lyrics: Much better handling of lyric text pieces, with the ability for the user to enter extension sign or to split words with a space. The OCR can process several text lines as a whole, which often leads to better results.
  - Symbols: The symbols bitmap definitions (in the /symbol folder) have been refined with at least a 16-pixel interline definition, resulting in better display notably in score view.
  - Tiff: Images are forwarded to Tesseract OCR by memory, avoiding temporary files

Explanation

1 Steps Internals How the engine works step by step.

TABLE OF CONTENTS

Steps internals

All the 20 steps of the OMR engine are to be presented here in chronological order.

This is a work in progress, to date only the first 9 have been documented in this handbook.

A step presentation is generally organized in:

1. goal
2. inputs
3. outputs
4. main processing operations

Some steps work on the sheet as a whole.
Other steps work system per system, and can present:

- sheet-level prolog
- system-level processing
- sheet-level epilog

> **NOTE**
> This type of documentation is halfway between a user manual (functional) and a developer manual (implementation). Reading it is not mandatory, but should help understand step by step how the engine behaves and how to best control it.

TABLE OF CONTENTS

- LEDGERS step
- HEADS step
- STEMS step
- REDUCTION step
- CUE_BEAMS step
- TEXTS step
- MEASURES step
- CHORDS step
- CURVES step
- SYMBOLS step
- LINKS step
- RHYTHMS step
- PAGE step

LOAD step

It is the very first step applied when processing any sheet. Its goal is to provide the sheet gray image, whatever the input color model.

Inputs

- The path to the book input file
- The sheet number within the book – 1 by default

Output

- The gray image of the sheet

Processing

1. A proper image loader is chosen according to the extension of the input file name (`.pdf`, `.tif`, `.png`, `jpg`, etc),
2. The loader then extracts the input image, based on the sheet number,
3. The image is discarded if its size is too large,
4. According to its color model, this initial image is converted to a gray scale of pixels – ranging from 0 for a black pixel to 255 for a white pixel.
    - The alpha channel if any is discarded,
    - If RGB channels are provided, the gray value for each pixel is set as the highest value among the 3 RGB values.

BINARY step

Many components of the Audiveris engine are meant to work on a black & white input image. In such image, the black pixels compose the foreground and the white pixels the background.

The goal of the BINARY step is to provide this black & white image.

Inputs

- The gray image of the sheet
- The chosen binarization filter – by default, this is the *adaptive* filter

Output

- The black & white image of the sheet

Processing

The gray value – a number between 0 and 255 – of each image pixel is compared with a threshold value. The result is:

- black (0) if gray value <= threshold value

- white (255) if gray value > threshold value

What is this *threshold* value? It depends on the binarization filter used on the sheet image.

Via the `Book → Set book parameters` pull-down menu, the precise filter and its parameters can be defined for the default, book and sheet scopes.

The GLOBAL filter

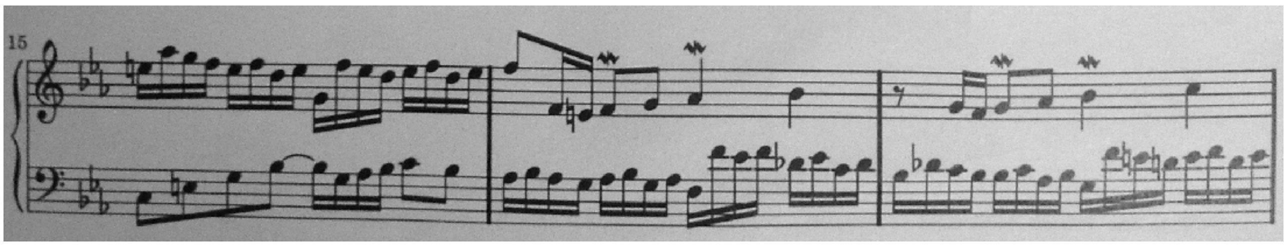
This is the older filter.

It provides a single threshold value which is applied globally on every pixel of the gray image.

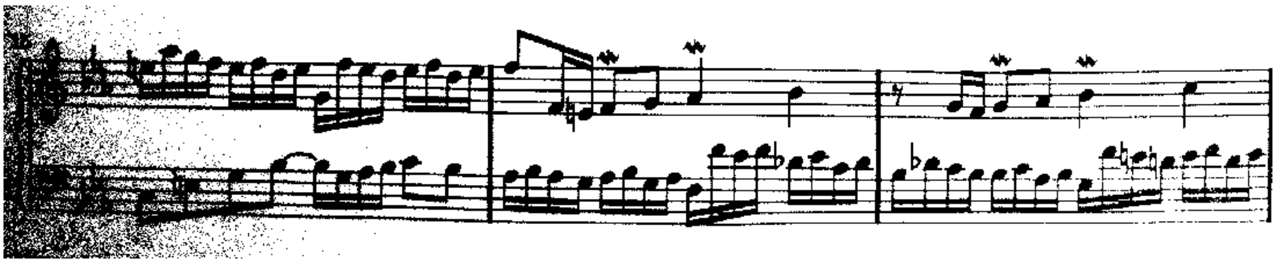
The threshold default value is `140`, and we can change this value via the `Book → Set book parameters` pull-down menu.

It generally provides good results, except when the input image exhibits different illumination values. Let’s consider the `BachInvention5.png` image, available in the `data/examples` folder of Audiveris.

Here is the bottom part of the gray image:



Note that the image appears dark on the left and pale on the right.
If we apply the *global* filter on it, we get this binary image:



The following engine steps will miss both the left part and the right part of this system. The left part because almost every pixel looks like foreground. The right part because several staff lines lack foreground pixels.

In this case, increasing or decreasing the overall threshold value has no point, because it cannot improve both parts at the same time.
We need something more sensitive to the local environment.

The ADAPTIVE filter

This is the newer filter.

Instead of being a constant value, a specific threshold value is now computed for every pixel, based on its location in the image.

The actual gray value is measured for every pixel in a rectangular neighborhood around the current location. Based on this population, the engine computes two values:[1]

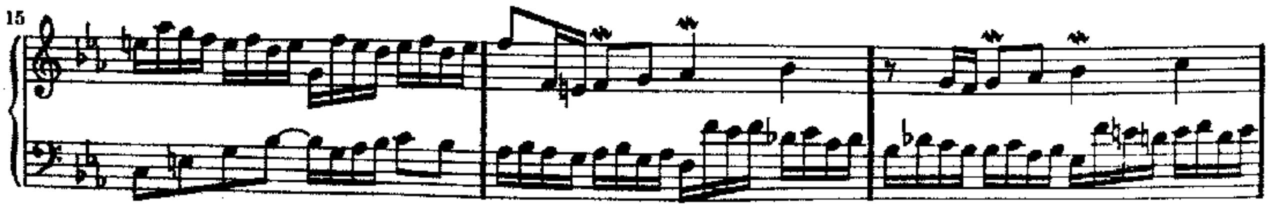
- `mean`: the mean gray value,
- `std_dev`: the standard deviation of gray values.

The resulting threshold is then defined according to the following formula:

> threshold = mean_coeff * mean + std_dev_coeff * std_dev

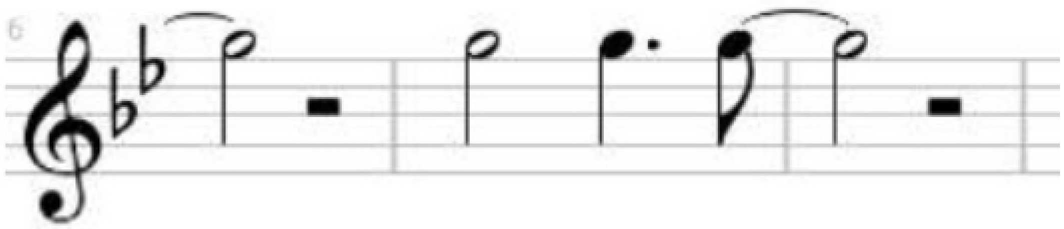
The parameters are by default `0.7` for `mean_coeff` and `0.9` for `std_dev_coeff`. We can modify them via the `Book → Set book parameters` pull-down menu.

On the same example, we now get this binary image:



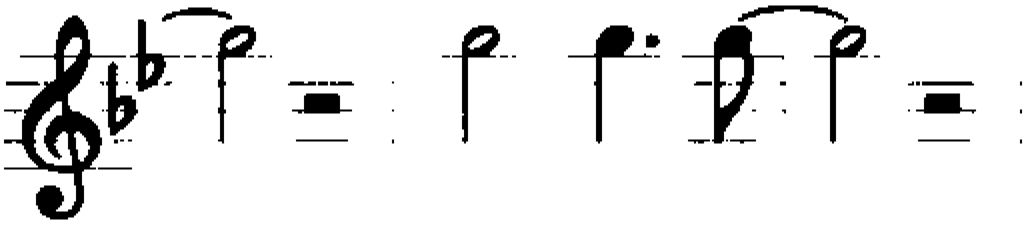
Our experience on hundreds of gray images has shown that the adaptive filter gives good results and its default parameters are OK. Therefore, the *adaptive* filter has been promoted as the default binarization filter for every input…

… until this discussion was recently posted on Audiveris forum.
Here below is a portion of the gray input image posted in the discussion:

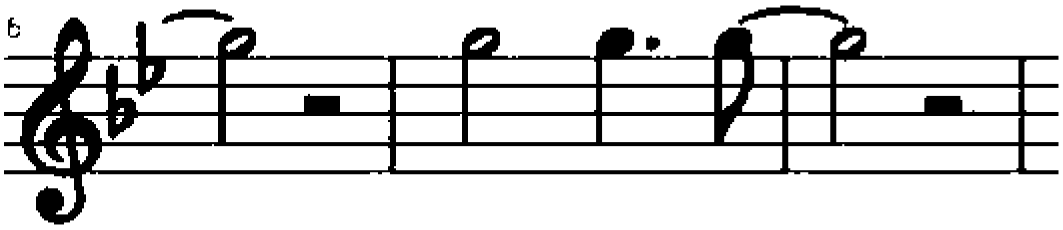


We can notice that the staff lines, as well as the bar lines, are painted in a pale gray color.
Here, the pale gray is not a transition between black and white as seen on the other scores.

Applying the standard *adaptive* filter gives this binary result:



Here, we must go back to the *global* filter.
With a threshold value set to 225, we now obtain this binary result:



Boards

Binarization board

The Binarization board is present in the `Binary` tab.

It allows the end user to interactively choose the filter, adjust its parameters, and observe the results on the image immediately.

Please refer to the Binarization board section.

Binary board

Available in the `Gray` and `Binary` tabs, the Binary pixel board presents the resulting black or white pixel value at the current location.

Please refer to the Binary board section.

Footnotes

[1] The implementation does not use brute force to compute mean and standard deviation around each and every point, but the result is the same. ↩

SCALE step

Music is written on staves composed of one or more horizontal lines regularly spaced within the same staff.

Depending on the input image, these lines can vary in thickness and spacing. And the dimensions of almost all the music elements to be detected depend on these values.

The purpose of the `SCALE` step is to measure this key scaling data for the sheet at hand.

Input

The black & white image provided by the `BINARY` step.

Outputs

Most of the scaling items use a tuple of 3 values, specified as a number of pixels:

- The minimum value
- The main (most frequent) value
- The maximum value

1. Mandatory outputs:

- Staff line thickness
- Staff interline – the vertical distance between staff lines, measured from line center to line center

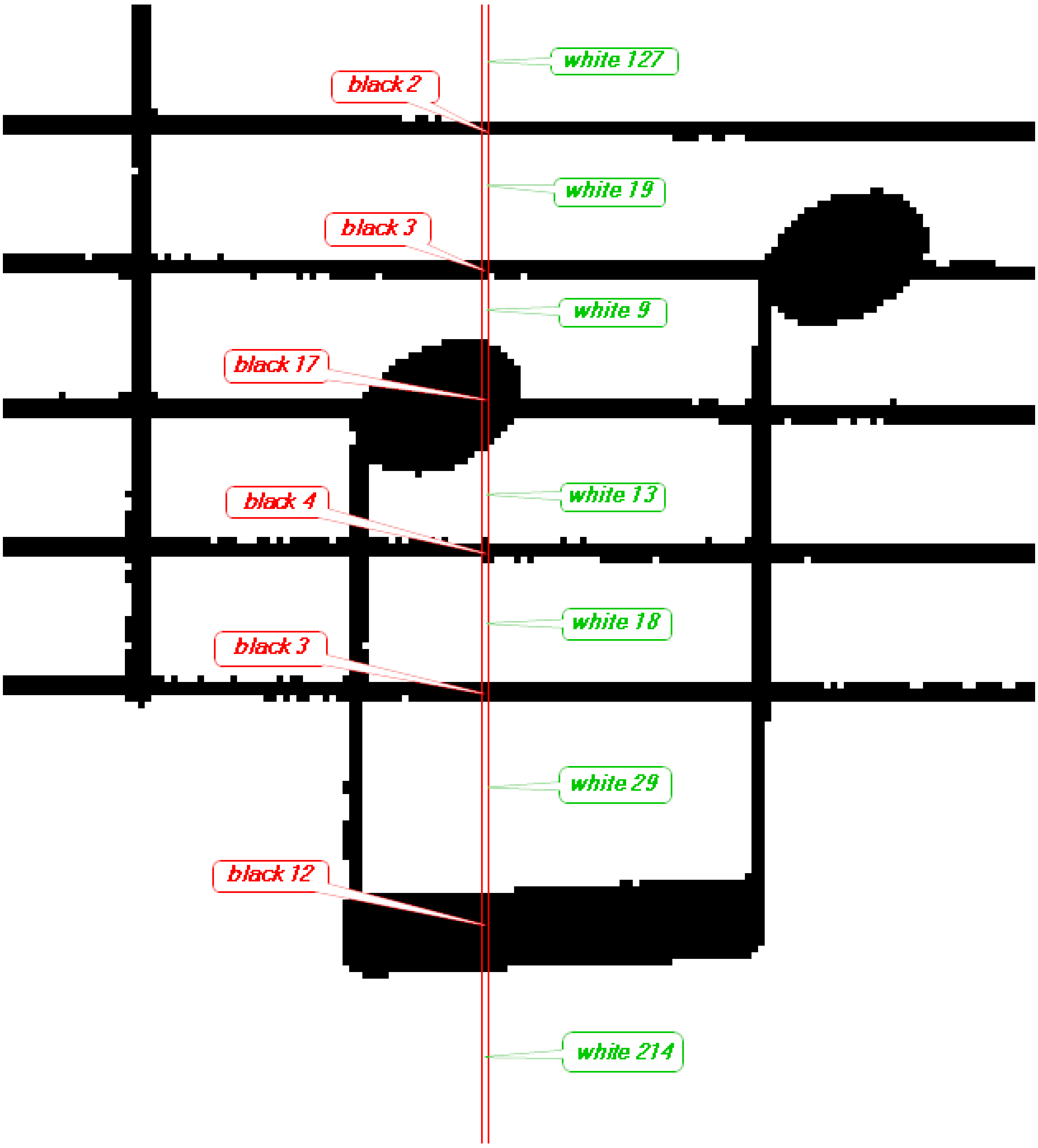
2. Optional outputs:

   - Small staff interline – when two populations of staves exist in the sheet, this is the vertical spacing for the smaller staves
   - Beam thickness – when beams exist in the sheet, this is the measured beam thickness
   - Small beam thickness – when two populations of beams exist in the sheet, this is the thickness for the smaller beams.

Processing

To retrieve the various scale values, the engine inspects every abscissa of the binary image for the sequence of vertical runs it contains, alternatively white and black runs.

The picture below was taken when working on the `chula.png` image found in the `data/examples` of Audiveris installation.
It is a partial view of a column read at a given abscissa, with the sequence of white and black runs and the length of each run in pixels:

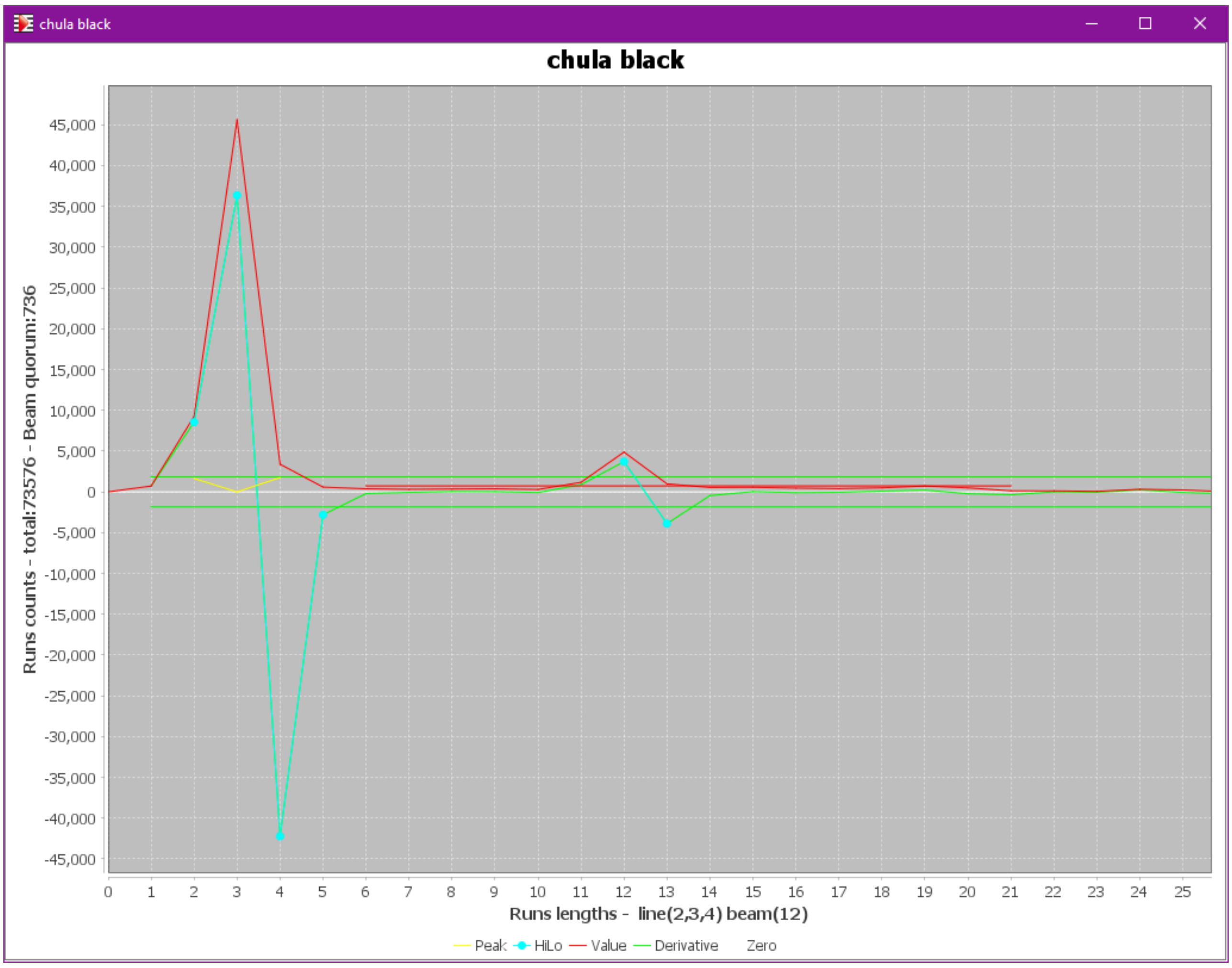


With all these vertical run lengths read in the whole image, the engine builds two histograms:

- The **black** histogram uses the length of every black run,
- The **combo** histogram tries to use the length of every white run twice
  1. combined with the length of the black run **above** if any
  2. combined with the length of the black run **below** if any

The final histograms can be viewed via the `Sheet → Display scale plots` pull-down menu – provided we have activated the `PLOTS` topic in the menu.

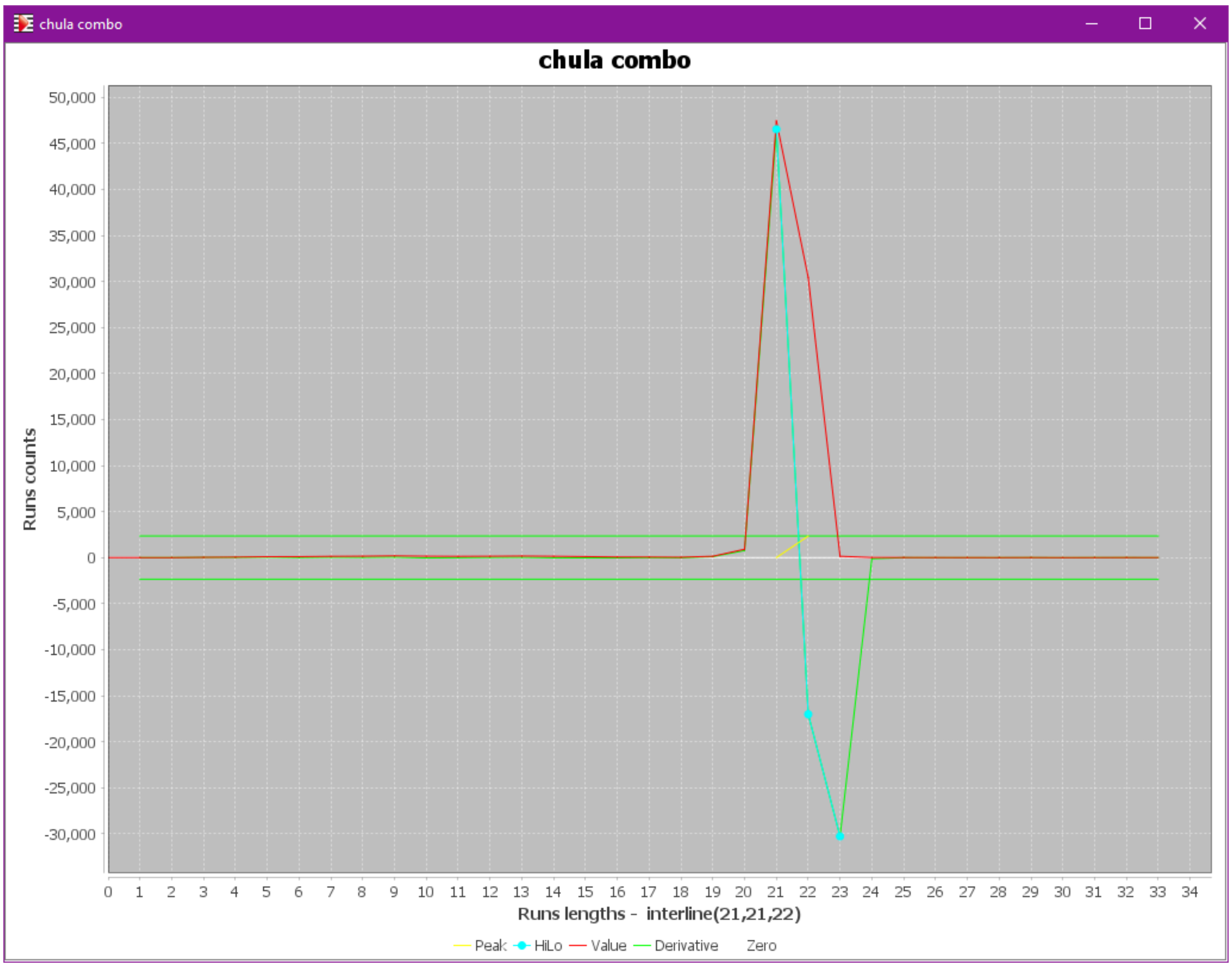
Black histogram



In this plot, we can clearly see two peaks on the length of vertical black runs:

- a first peak around value 3, this is the typical staff line thickness
- a second peak around value 12, this is the typical beam thickness

Combo histogram



Here we have just one peak around value 21, this is the typical interline value.

Engine behavior

The engine detects these peaks rather easily, using:

- a threshold on peak height (for beam thickness in the black histogram)
- a threshold on the absolute values of derivative before and after the peak (leading to the “HiLo” sequences shown on the plot), to precisely grab the minimum and maximum values around the peak.

In the current example, the output of the `SCALE` step can be read in the log as:

> [chula] Scale{ interline(21,21,22) line(2,3,4) beam(12)}

This simple example represents the most frequent case.
We can also encounter more complex cases as listed below.

Regarding the *black* histogram:

- It may exhibit just one peak, meaning that no significant beam-based runs were detected. The image can in fact contain no beam, but it can also contain too few of them, resulting in no clear peak.

In the latter case, the end user may have to explicitly provide the beam thickness to the engine. See this sheet scale section about beam thickness.

- It may exhibit 3 peaks, one for the typical line thickness and the two others for two populations of beams: small and large (standard).

Regarding the *combo* histogram:

- It may exhibit 2 peaks, due to the presence of 2 populations of staves in the image, one with a smaller interline and one with a larger (standard) interline.
- It may exhibit no clear peak, or a peak detected far from the expected values – Audiveris expects an interline value not too far from 20 pixels.
  - This generally means that the image at hand contains no staff, but perhaps some illustration. The engine will notify the user about a sheet considered as “*invalid*” from the OMR point of view.
  - This case can also occur when the score contains no multi-line staff, hence there is no physical *inter*-line to measure. See snippets with only one-line staves.

GRID step

The purpose of the `GRID` step is to retrieve the systems and staves of the sheet.

Inputs

- The black & white image
- The maximum staff line thickness
- The typical interline value (and the small interline value if any)

Outputs

- The LAG of horizontal sections – See Section and LAG description.
- The LAG of vertical sections
- The staff lines organized in staves
- The global skew of the image
- For every staff:
  - the barlines if any
  - the vertical connectors if any with a staff above or below
  - the starting brace if any

    - the starting bracket if any
- The gathering of staves into systems
- The no-staff image

Processing

Filtering runs and sections

First, all the foreground (black) pixels of the binary image are organized into vertical runs.

Then, the engine determines a maximum line thickness, based on the maximum staff line thickness provided by the `SCALE` step, slightly augmented to cope with potentially thicker ledgers.

Any vertical run which is longer than this maximum staff line thickness cannot be a pure portion of a staff line – it must be a (portion of) crossing object.

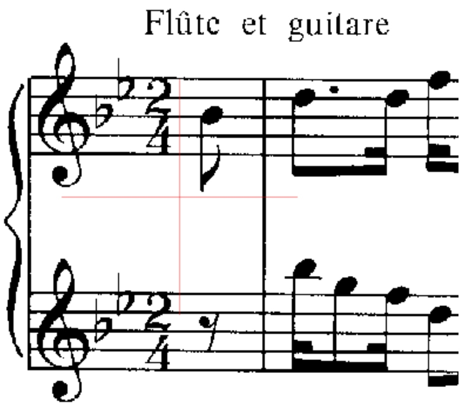
Therefore:

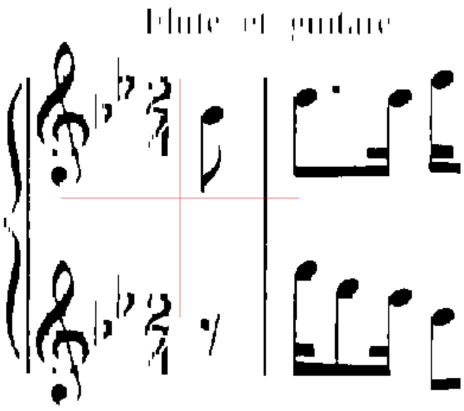
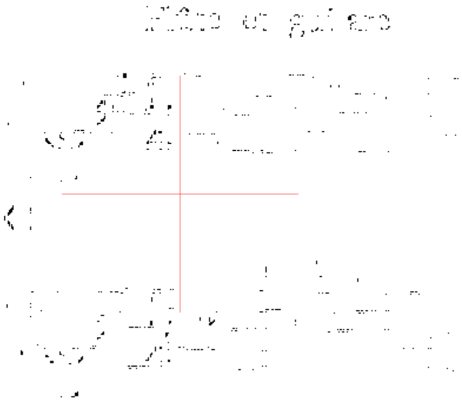
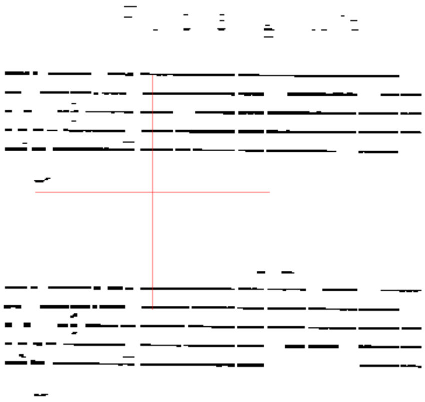
- All “long” vertical runs are set aside into the *vertical LAG*.
- All remaining pixels are organized in horizontal runs gathered in the *horizontal LAG*.

Then, within each LAG, adjacent runs are organized into sections.

The horizontal sections are further separated between short sections and long sections, because the long sections will drive the staff lines detection.

We can visualize this material. To do so, we have to use the `Tools → Constants` pull-down menu and there tick the boolean constant `displayRuns` (it is located in class `LinesRetriever`).

Content	View
Binary image	

Content	View
Long vertical sections	
Short horizontal sections	
Long horizontal sections	

Long horizontal filaments

From the set of long horizontal sections, the engine gradually builds long and thin filaments which are likely to correspond to long portions of staff lines.

The filaments that are not straight enough are discarded. A small set of filaments, the longest ones, are used to compute a global slope of the score image. The filaments whose slope diverges from the sheet global slope are discarded.

Then the horizontal filaments are vertically gathered into clusters likely to represent staves. Within a cluster:

- the filaments are expected to be spaced according to a known staff interline

- the number of filaments must match the possible staff sizes (1-line, 4-line, 5-line or 6-line) as defined by the end user via the `Book → Set book parameters` pull-down menu.

In the end, the remaining clusters are transformed as standard staves, one-line staves or tablatures according to their line count.

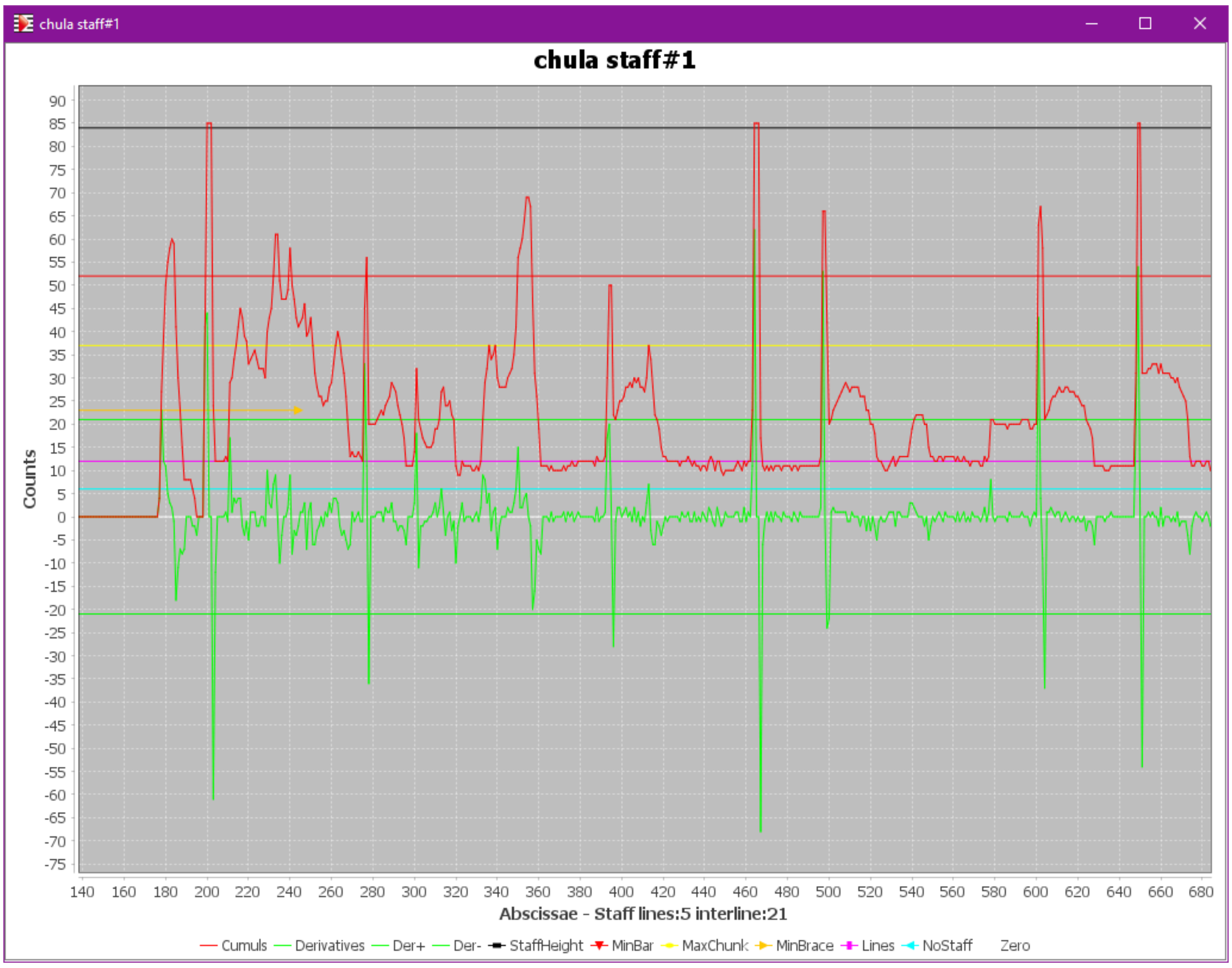
At this point, the left and right abscissa limits of each staff are defined purely via its detected lines.

Staff vertical projection

Since the engine knows the top line and the bottom line for each staff, it can compute a projection of all black pixels located between these two lines onto the x-axis.[1]

Provided we have activated the `PLOTS` topic in the pull-down menu, we can visualize this projection:

- either via the `Sheet → Display staves plots` pull-down menu
- or via the `≡ Staff N°` contextual menu from a staff



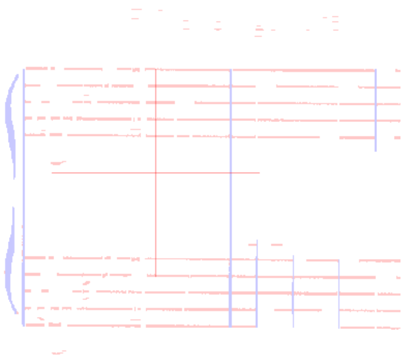
From this projection limited to the staff height, the engine detects peaks which can represent:

- barlines,
- half braces,
- half brackets,
- as well as long stems, etc.

The slim vertical sections located within a peak are used to build vertical filaments. Note that these filaments may extend past the top and/or the bottom of the staff.

If, via the `Tools → Constants` pull-down menu, we have ticked the boolean constant `displayFilaments`, then a specific `Filaments` tab is displayed for both horizontal and vertical filaments initial material:

- The horizontal filaments are in pale red
- The vertical filaments are in pale blue

Content	View
Initial Filaments	

All the peaks in vertical projection are handled as vertices in a common peak graph at sheet level. Any vertical alignment detected between a peak in one staff and a peak in the following staff is recorded as an edge in the global peak graph.

Each alignment is then checked for the presence of enough foreground pixels between the two staff peaks, to detect concrete connections between them.

Any filament which extends beyond the staff top or bottom limits and which is not a connector cannot be a barline and is thus discarded.

Based on the detected connections, the engine is now able to gather staves into systems.

System internal alignments

Within each system, the staff peaks are organized into system-based columns.

The starting column, if any, is used to refine the system and staves left abscissa.

The columns found on the right of the starting column must embrace the whole system height, otherwise they are discarded. Also, the peaks that extend past the system vertical limits are discarded.

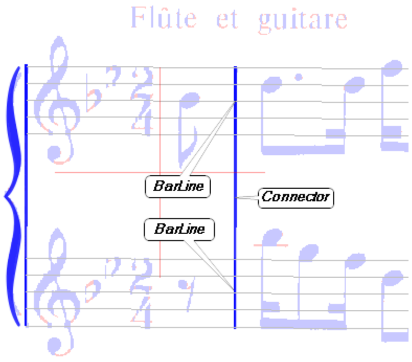
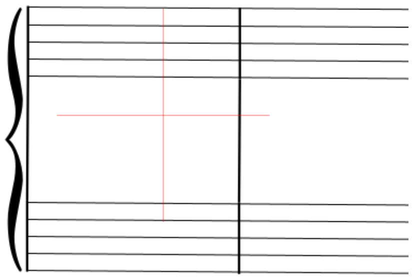
Brace portions and bracket portions are searched on the left of the starting column. The presence of braces in left margin drives the gathering of staves into parts.

Finally, Inters are created in each system SIG for:

- barlines
- connectors
- braces
- brackets

Results

At the end of this `GRID` step, we can see the `Data` tab populated with the grid of systems and staves (shown below in physical and logical modes):

Mode	View
♫ Physical	
♫ Logical	

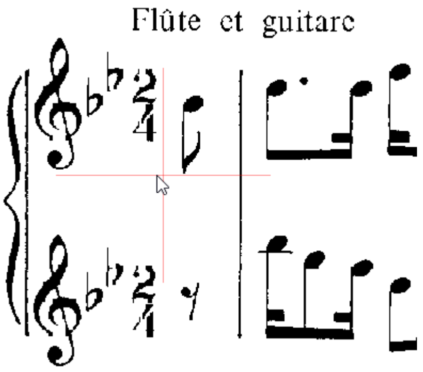
Note that some false barlines may persist at the end of the `GRID` step. They generally get discarded later in the `REDUCTION` step.

The no-staff image

Since the engine knows which precise horizontal sections compose the staff lines, it can logically “erase” these sections from the binary source and provide what is called the no-staff image.

The no-staff image will be used by several steps down the pipeline.

We can visualize the no-staff image via the `Sheet → Display no-staff` pull-down menu:

Content	View
No staff	

Footnotes

[1] In the specific case of a 1-line staff, the engine uses a target barline-height defined as a number of interlines (and perhaps modified via the `Book → Set book parameters` pull-down menu). ↩

HEADERS step

The `HEADERS` step detects clef, key and time information at the beginning of every staff.

Inputs

- The systems and staves
- The no-staff image

Outputs

For every staff:

- The clef (mandatory)
- The key signature, if any
- The time signature, if any

Processing

The engine can be rather aggressive because of the standardization of the sequence of these entities in the header portion of the staff.

Moreover, within the same system, the header components (whether present or absent) are vertically aligned in “columns” across the system staves. This allows several cross-validations within a multi-staff system:

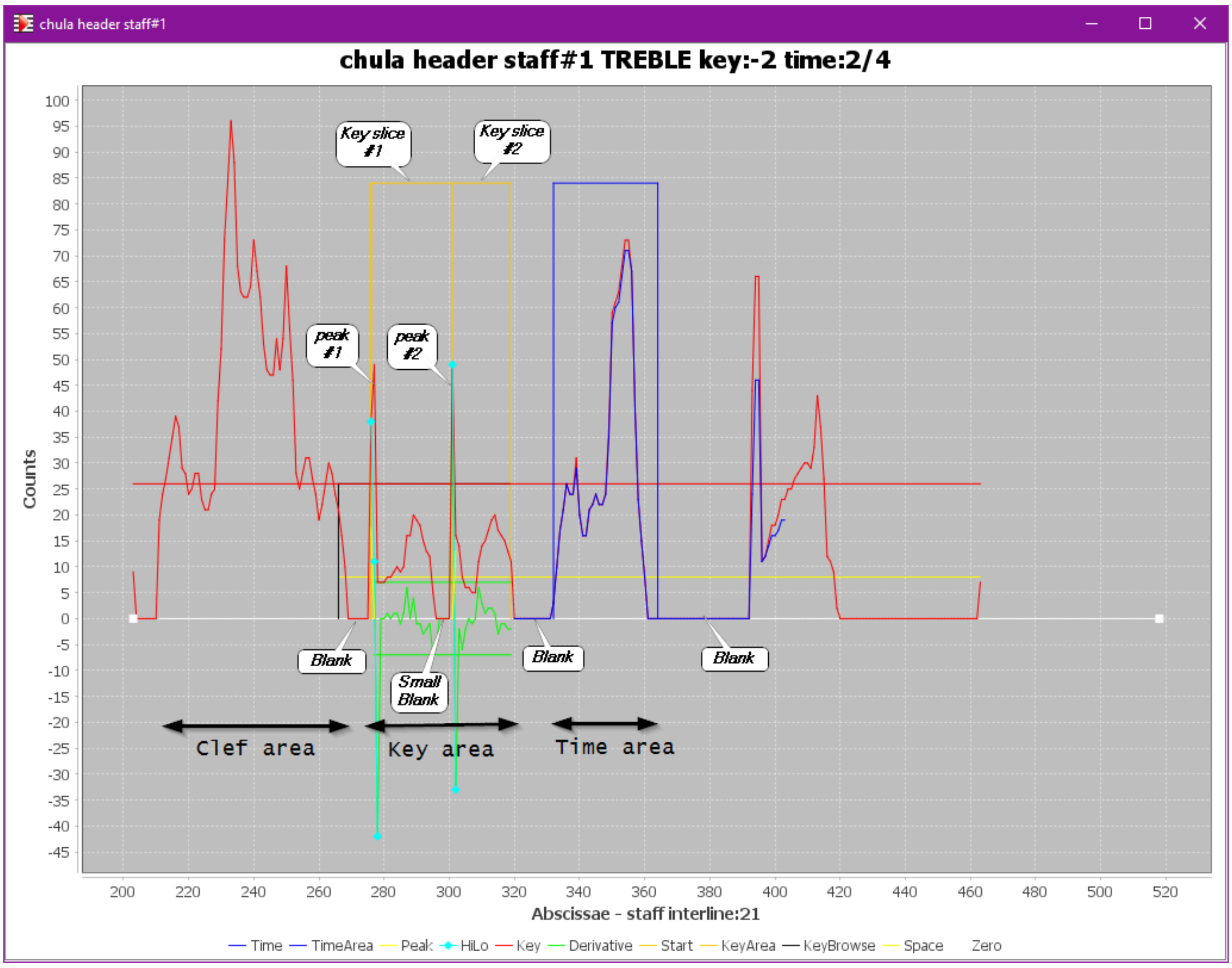
- **Clefs**:
  A clef is mandatory at the beginning of each staff header.
  It can be different from one staff to the other.
- **Key signatures**:
  A key signature can vary (and even be void) from one staff to the other.
  If present, the alteration slices are aligned across system staves.
- **Time signatures**:
  Either a time signature is present and identical in every staff of a system, or there are none.

Staff-free pixel source

To process each staff header, the engine uses the no-staff image – the binary image where all staff line pixels have been removed, made available by the `GRID` step.

With a projection to the x-axis, this allows to detect the blanks between the header components and thus limit the region of interest for each of them.

Specifically for the key signature, the projection is also used to detect peaks likely to represent “stem-like” portions of sharp or flat signs.



Clef candidates

Using the pixels of a rectangular lookup area, the engine extracts all the elementary glyphs – defined as connected black pixels ensembles.

These elementary glyphs, as well as their combinations, are submitted to the glyph classifier, to come up with a list of acceptable clef shapes.

There may be several (mutually exclusive) acceptable clefs. They are registered in the system SIG and kept for the time being.

Key candidates UPDATED IN 5.8

A key signature is a sequence of consistent alterations (all sharps or all flats or none) in a predefined order (FCGDAEB for sharps, BEADGCF for flats).

WARNING

In the case of a key signature change, there may be some natural signs to explicitly cancel the previous alterations, although this is not mandatory. As of this writing, the engine is not able to handle key signatures containing natural signs.

The relative positioning of alterations in a given signature is identical for all clefs (treble, alto, tenor, bass) with the only exception of the sharp-based signatures in tenor clef.

Regarding the vertical projection of the staff-free pixels onto the x-axis:

- Vertically, the projection uses an envelope that can embrace any key signature (under any clef), ranging from two interline values above the staff to one interline value below the staff.
- Horizontally, the goal is to split the projection into slices, one slice for each alteration item to be extracted.

At the staff level, the engine strategy is as follows:

1. Find the first significant space right after the clef, it’s the space that separates the clef from the next component (key signature or time signature or first note/rest, etc). The next really wide space will mark the end of the key signature area.
2. Look for projection peaks in the area representing “stem-like” portions (one for the flat shape, two for the sharp shape)
   Since at this point in time there is no reliable way to choose between flats and sharps, the engine will work in parallel on the two hypotheses (flat-based key vs. sharp-based key).
3. Determine a precise splitting of the projection into vertical regions of interest (slices), based on the “stem-like” peaks and the shape hypothesis.
4. Try to retrieve one good component in each slice, by submitting each glyph compound to the glyph classifier. For the slices left empty, force slice segmentation and perform recognition within the slice only.
5. Check each item pitch against the pitches sequences imposed by the staff clef candidate(s). In the system SIG, register a support relationship between any compatible clef candidate and key candidate.

And for the systems that contain several staves:

1. Across all the system staves, align the abscissa offset for each slice.
2. Check that within any multi-staff *part*, the part staves exhibit the same key signature, otherwise select the best one and replicate it in the other staves of the part.

Time candidates

Since all time signatures must be identical in any vertical “column” of a system, the retrieval is made directly from the system level:

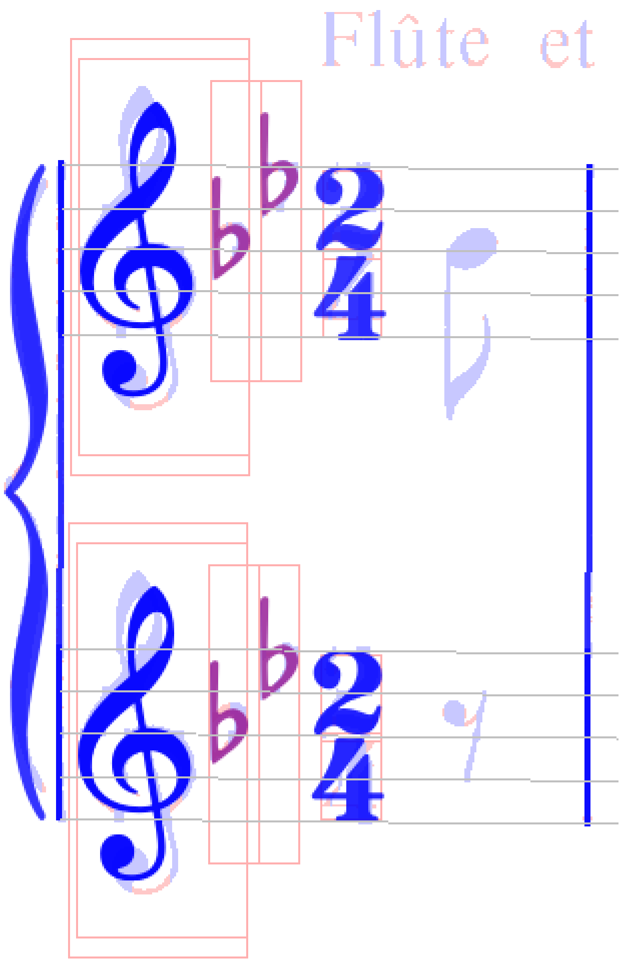
1. In the system header area, a “column” of staff areas is defined, right after the key “column” if any, otherwise right after the clef “column”.
2. For each staff of the system, knowing the abscissa range for time candidates, three lookup regions are defined:
   - A rectangle as high as staff height, searched for possible whole time signature shapes such as COMMON_TIME, 4/4, 6/8, etc.
   - A small rectangle centered on the higher staff half, searched for possible numerator shapes
   - A small rectangle centered on the lower staff half, searched for possible denominator shapes
3. Not all num/den pairs are possible, the engine restricts them to the commonly used ones.
4. If, in a staff header, no time signature was found, the search within the current system is abandonned.

Results

The picture below presents the results of the `HEADERS` step on the first system of the `chula` input example.

We can see the lookup areas used for:

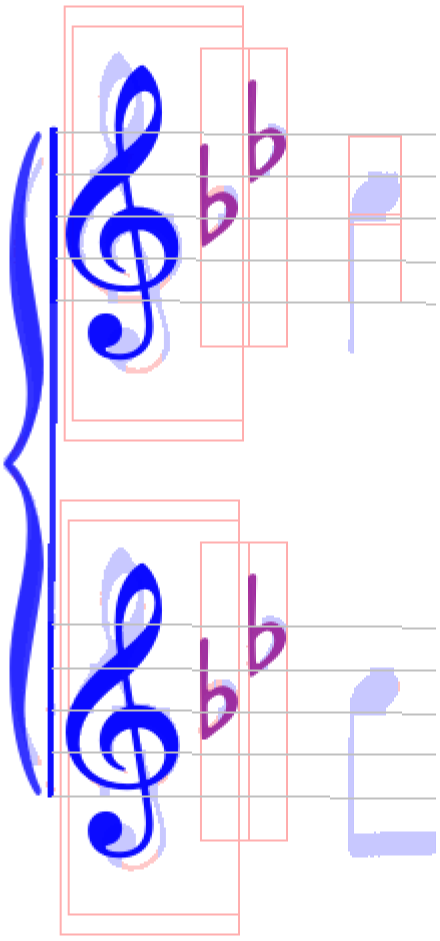
- Clef: outer and inner rectangles
- Key: rectangular slice for each item of this two-sharp key
- Time: one whole and two half rectangles



The next picture presents the results on the second system.

Notice that, in the first staff, the projection of the quarter note, located right after the key signature area, has been searched for a time signature candidate. But no time symbol could be recognized there.

Consequently, the header of the second staff was not even considered for a time signature lookup.



STEM_SEEDS step

A stem is a rather long vertical segment meant to join note heads, perhaps with beams. And in a perfect world, any stem could be easily recognized on its own.

But in reality, there may be missing black pixels in the input image, perhaps breaking a stem in smaller segments, or damaging the connection between a stem and some of its partnering heads and beams.

To cope with this, the engine uses several approaches to gradually detect:

1. (portions of) stem candidates – this `STEM_SEEDS` step!
2. beam candidates – the `BEAMS` step
3. ledger candidates – the `LEDGERS` step
4. heads candidates – the `HEADS` step

and finally to:

1. determine all full stem candidates joining heads and beams – the `STEMS` step
2. filter all the candidates – the `REDUCTION` step

Input

- The systems and staves
- The no-staff image
- The vertical LAG

- The horizontal LAG (very partially)

Output

- The typical stem thickness for the sheet
- The stem seeds as glyphs in the `VERTICAL_SEED` group

Prolog: stem thickness

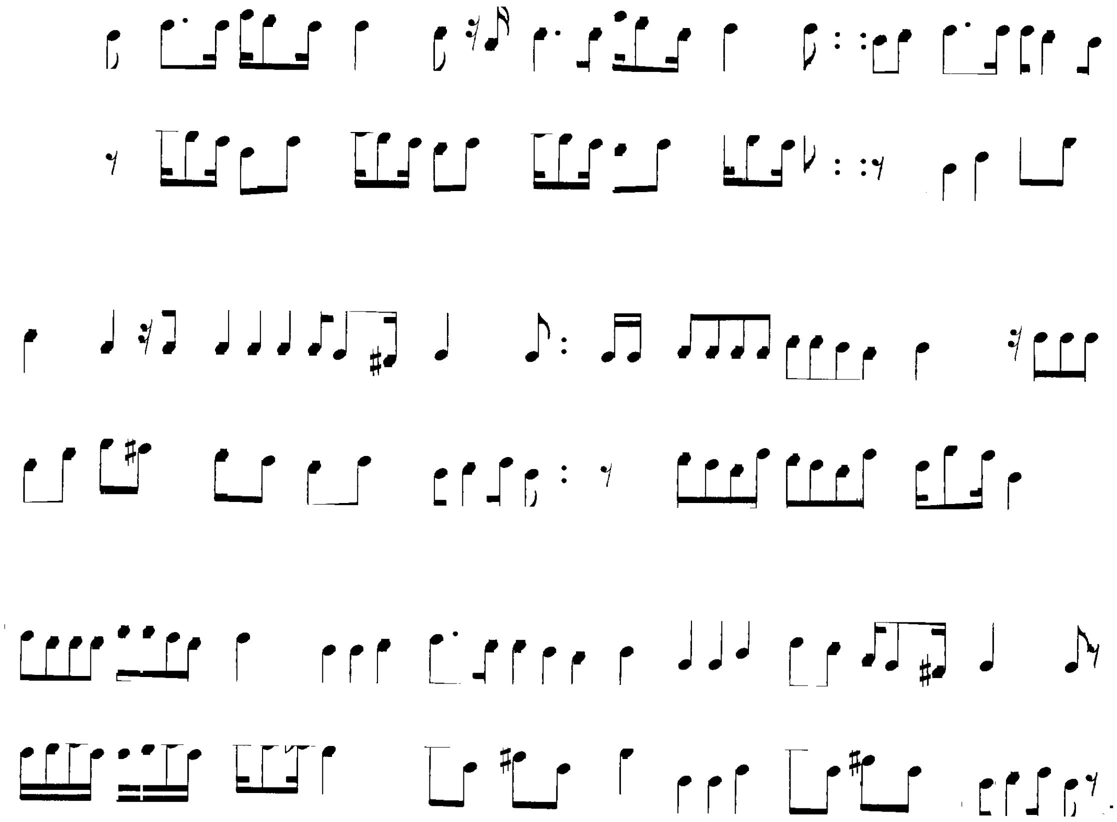
The user can directly specify a stem thickness value for the sheet at hand via the `Sheet → Set scaling data` pull-down menu.

In the normal case – that is with no thickness value provided – the engine must, as a step prolog, measure the typical stem thickness. It does so by crafting an input image suitable for a global stem analysis.

To avoid any pollution brought by lyrics, title or other external Main concepts, the engine focuses only on the core staff areas, which is the staff height augmented vertically slightly less than one interline above and one interline below.

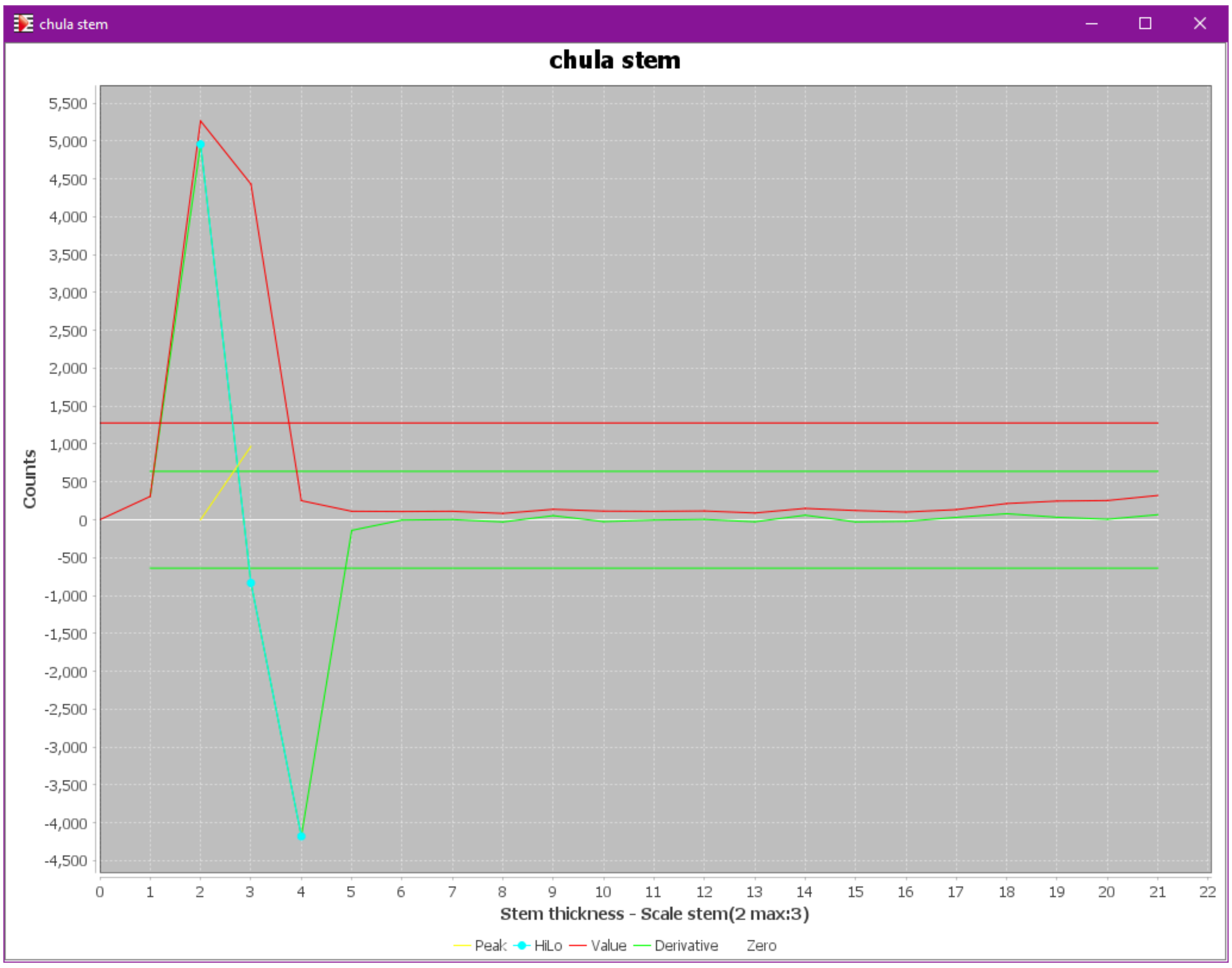
Within each staff area, the staff lines have been removed, as well as the detected barlines and the header components.
Note that some stems may have been mistaken for barlines candidates during the `GRID` step; this should have a negligible impact on the stem thickness computation.

The result is an input image like this one (still for the `chula` example):



On this input, it is easy to build a histogram of horizontal black runs lengths, as shown below.

We can display this histogram via the `Sheet → Display stem plot` pull-down menu – provided we have activated the `PLOTS` topic in the menu.



The retrieved peak gives the stem thickness: in our example, the most frequent value is 2 pixels, and the maximum value is 3 pixels.

These values are stored in the sheet scaling data. Then the rest of step processing is performed system per system.

System processing

Vertical filaments

Based on the retrieved (or specified) stem thickness value, the engine can now build vertical straight filaments from the vertical sections that are slim enough and long enough.

For now, the engine is only looking for stem *seeds*. Hence, it makes no attempt to extend these filaments vertically – this will be done later in the `STEM` step.

The filaments can just be thickened with compatible adjacent vertical sections and tiny horizontal sections as long as the resulting thickness stays within the limit on stem thickness value.

Checking seeds

Each retrieved filament is further checked to be kept as a stem seed.

The checks concern:

- slope
- straightness
- length
- cleanliness – not too many adjacent pixels
- high black part
- low white part

The checks are evaluated according to the declared quality of the sheet input (synthetic, standard, poor).

The successful candidates are finally registered in the specific `VERTICAL_SEED` group of glyphs, ready to be picked up by the following steps.

BEAMS step

The purpose of the `BEAMS` step is to detect all beams in a sheet, both (standard) beams and small beams if any.

As a side effect, it also detects any multi-measure rest found in the sheet.

Inputs

- The no-staff image

Outputs

- The beams and beam-hooks candidates
- The multi-measure rests candidates

Prolog: retrieval of potential beams glyphs

Geometrically, a beam is a filled parallelogram, rather horizontal, long and thick.

To focus as much as possible on all relevant beam candidates and only them, the input data is carefully worked upon.

The engine cleaning actions are as follows:

1. Remove all the staff lines, by taking the no-staff image as input
2. Remove all stem-like items, by removing any horizontal run whose length is smaller than the maximum stem thickness
3. Smoothen the image to get rid of noise and small holes, by applying a median filter followed by a gaussian filter, both using a 3x3 kernel
4. Highlight the spots whose thickness is close to the beam expected thickness;
   this is the result of a morphological closing operation (a dilation followed by an erosion) using a disk-shaped mask whose diameter is the expected beam thickness – the smaller value, when the sheet exhibits two beam thickness values

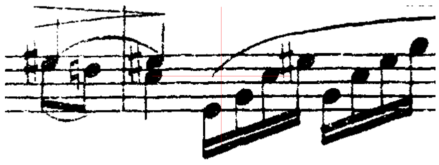
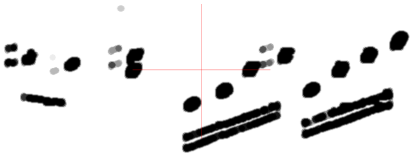
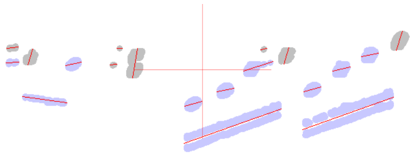
The resulting cleaned image uses a gray scale.
All spots with a rather high density of black are interesting for two purposes:

- beams recognition of course,
- but also black heads later recognition by the `HEADS` step. [1]

Regarding beam recognition, the prolog final actions are as follows:

1. Binarize the resulting gray image, using a global threshold
2. Build all glyphs (spots) from the image runs, using runs connections
3. Dispatch all spots to their containing systems – any spot located between two systems is dispatched to both

As an example, the images below result from the processing of a score of rather poor-quality.
Notice that, in the cleaned image, the tiny black or white items have disappeared.

Legend	Image
Binary image	
Cleaned image	
Detected glyphs	

The `binary` tab is always available via the `Sheet → Display binary` pull-down menu. The two other tabs require specific booleans to be set in the `Tools → Constants` menu, respectively `displayGraySpots` and `displayBeamSpots`.

Beams detection

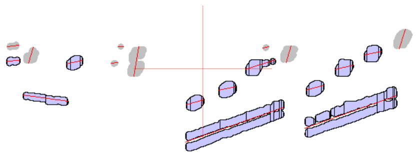
Each individual spot glyph is checked for being a good beam candidate:

- width
- mean thickness
- slope of the glyph center line (least squares approximation)

These very simple criterias help discarding some glyphs (displayed in gray in the image above).

At this point, a glyph may represent a beam, or a portion of a beam, or several beams stuck together, or a black note head, etc…

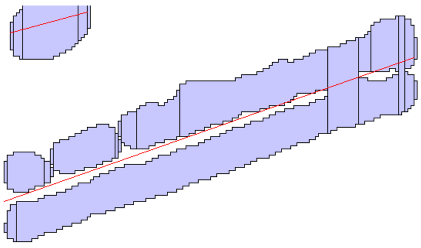
For each of the remaining glyphs (displayed in pale blue), the engine builds a structure, based on the glyph vertical sections:

Legend	Image
Sections	

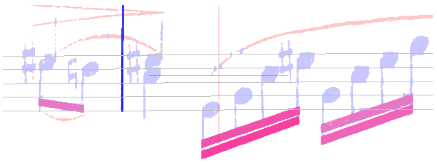
The structure focuses on the top and bottom lines of the most significant vertical sections – the `View → Switch selections` menu allows the display of sections.

These lines are likely to represent the top and bottom borders of the underlying individual beams. They are checked for straightness and consistent slope.

A thick structure is likely to represent beams stuck one upon the other, it is thus split into thinner structures:

Legend	Image
Double beam glyph	

The final structures are converted to elementary beams, which can be further merged or extended horizontally to full beams and beam hooks Inters.

Legend	Image
Beams inters	

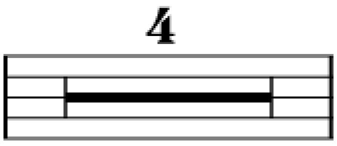
At the end of this `BEAMS` step, all the created Inters are just candidates. In particular, Inters for short beams and beam hooks may still coexist for the same underlying glyph!

These beam Inters are displayed in red: they are still in an *abnormal* status becaused they are not yet linked to any stem!

Only the following steps, especially the `REDUCTION` step, will allow the engine to confirm or discard these candidates.

Multi-measure rests detection

Here is a multi-measure rest example:



It looks like a horizontal beam, vertically centered on the staff middle line, and with vertical serifs on each horizontal side.
A number (“4” in the example), located above the staff, indicates the number of measures covered by the multiple rest.

At the end of the `BEAMS` step, every beam candidate is checked to decide if it should be considered as a multi-measure rest rather than a plain beam.

The checks are rather obvious:

- Minimum length
- Maximum vertical distance from staff mid-line for the left and right extremum points
- Presence of vertical serifs on left and right sides. At this point in time, the stem seeds have not yet been detected, therefore the engine uses a staff projection to detect peaks and thus serifs on both sides.

NOTA: The count above the staff cannot be detected during the `BEAMS` step. The `SYMBOLS` step should detect this number symbol, and the subsequent `LINKS` step should link the multi-measure rest and the count via a MultipleRestCountRelation.

Footnotes

[1] This is a forward reference to the `HEADS` step, since filled black heads are very likely to appear in these spots. A specially binarized version of the cleaned image is thus set aside to be used later by the `HEADS` step. ↩

LEDGERS step

The `LEDGERS` step tries to detect all ledgers in a sheet.

This data is important because the following scan for heads will be driven precisely along and between the staff lines and ledgers of the sheet. See the `HEADS` step.

Inputs

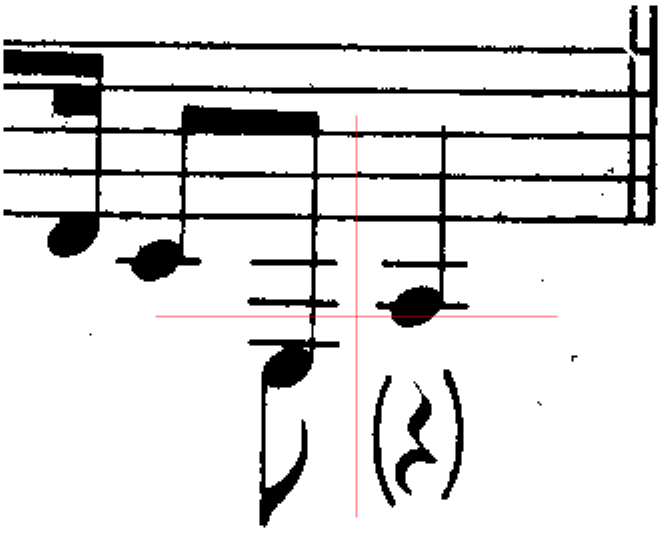
- The no-staff image
- The detected staves

Outputs

- The collection of ledgers, organized by (even) pitch around each staff

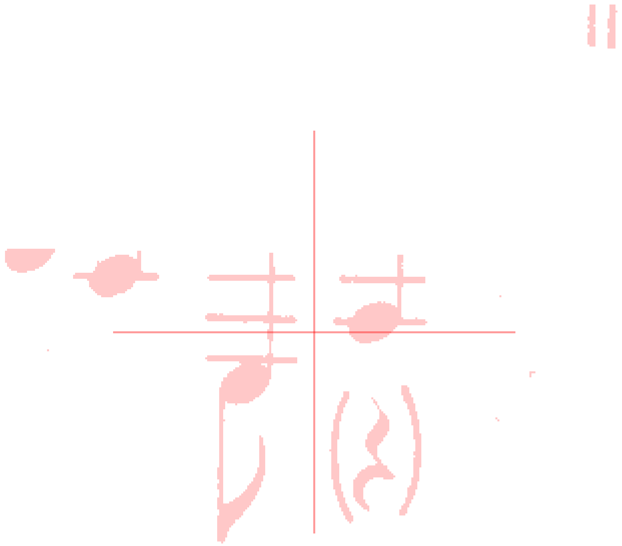
Prolog: retrieval of candidate sections

In terms of vertical distance, a ledger cannot be closer than one interline to a staff.



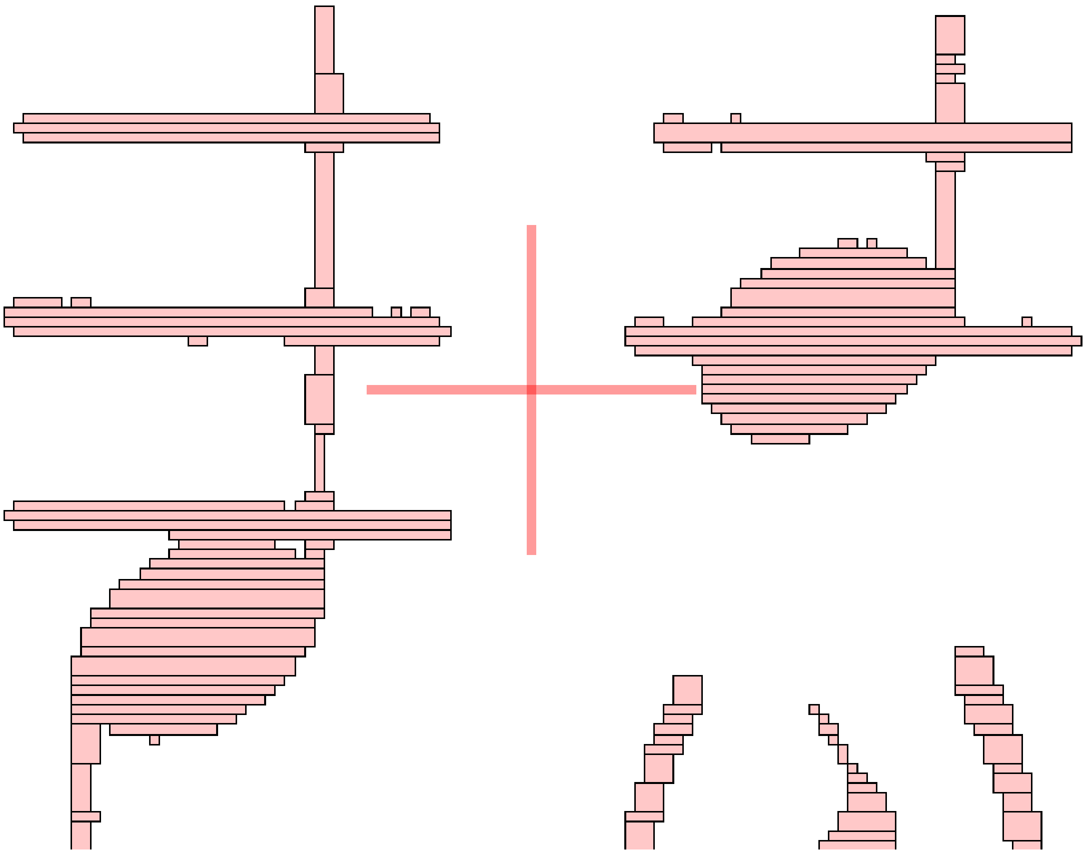
Therefore, the engine takes the no-staff image as input, and erases the areas occupied by the detected staves, vertically augmented by a half-interline margin above and below each staff.

This gives the picture below, where the remaining pixels are viewed as a horizontal run table, hence the pale red color.



These runs are then gathered into horizontal sections. The chosen policy is to combine adjacent runs only when they exhibit the same starting and ending abscissa values.
This results in rectangular sections as can be seen below in a magnified view of the `Ledgers` tab:

(the `Ledgers` tab is displayed only if the constant `displayLedgers` is set to true beforehand)



These horizontal sections are finally dispatched to their related systems (if a section is located between two systems, it is dispatched to both).

Ledgers detection at system level

The engine uses a stick factory to build short straight horizontal filaments. (the same as used vertically in the `STEMS_SEEDS` step).

There is no possible histogram approach to measure the expected ledger thickness. And it can be as thick as 3 times the staff line thickness. So, the engine uses a maximum value extrapolated from the staff line thickness and the staff interline value.

Filaments that are too long or which intersect (good) beam candidates are discarded.

The candidate ledgers are then checked incrementally around each staff:

- Starting one interline above staff and going up, one interline at a time,
- Starting one interline below staff and going down, one interline at a time.

And of course, to be considered, a candidate ledger must build upon the needed intermediate ledgers.

A candidate ledger is then applied weighted checks:

- Minimum length
- Minimum and maximum thickness
- Straightness
- Gap with the theoretical vertical location
- Convexity on horizontal sides (check for white pixels above and below the ledger end portions)

If the resulting check grade is too low, the ledger candidate is discarded (and no further candidate can be looked up beyond the candidate position).

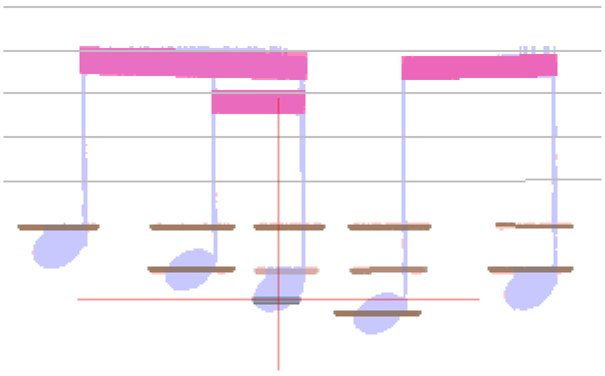
Epilog: post-analysis

During the collecting of ledgers candidates at system level, the engine had only empirical values of acceptable ranges for ledger attributes.
Moreover, we must remember that this `LEDGERS` step is performed before the `HEADS` step. So, at this point in time, the engine cannot use any information coming from head candidates.

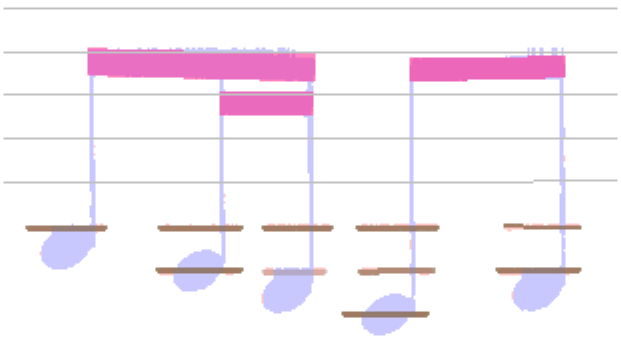
But now that the whole population of ledgers candidates is available for the current sheet, the engine can compute the mean value and the standard deviation for key attributes and re-check each ledger candidate about:

- the vertical distance from the previous ledger or staff line
- the ledger thickness

The purpose of this post-analysis is to further filter out some candidates.
For example, in the picture below, the designated candidate is somewhat too close vertically to the ledger above it:



The additional filtering leads to this final configuration:



Remark: This is not the end of it, since the `REDUCTION` step, which is run after the `HEADS` and `STEMS` steps, will discard the ledgers left unused.

HEADS step

The HEADS step uses a template matching technique to retrieve all note head candidates (black head, void head, whole head) in the current sheet.
The note rests will be addressed later in the SYMBOLS step.

Inputs

- The BINARY image of the sheet
- The interline value for each staff
- The staff lines and ledgers
- The stem seeds
- The book parameters (musical font family, specific staves, specific head shapes)

Outputs

- The detected note heads

Prolog

Building the distance table

On a copy of the `BINARY` image, all the pixels that compose horizontal lines (staff lines, ledgers) as well as vertical lines (barlines, connectors and stem seeds) are “erased”, that is replaced by the background color (white).

Based on the resulting image, the engine computes the sheet “distance table”: Each (x,y) cell of this table contains an integer value, which represents the euclidian distance from the (x,y) pixel to the nearest foreground (black) pixel. [1]

For the “erased” elements (the vertical and horizontal lines), the distance integer value is replaced by a special value (-1) to indicate that these locations must be ignored by the template matching process. The reason is these black pixels in the input image would still be there, regardless of the presence or absence of any head.

A cell of the distance table can now contain:

- the value `0` (foreground point)
- a positive value (distance of this background point to the nearest foreground point)
- the value `-1` (point to be ignored)

Loading the head spots

The engine loads the “head spots” detected and stored during the `BEAMS` step.

These spots are likely to correspond to black heads and are therefore used to speed up the application of black head templates along a staff/ledger line even without stem seeds.

System processing

The processing is done separately for every staff of every system.

The engine considers a collection of possible head shapes (out of a total of about 25) which can be narrowed down depending on the selection made in the `Book → Set book parameters` menu, notably:

Option	Expected note heads
[X] 1-line percussion staves	Oval
[X] 5-line standard staves	Oval
[X] 5-line unpitched percussion staves	Drum notation
[X] Small heads	Small oval
[X] Cross note heads	Crossed oval

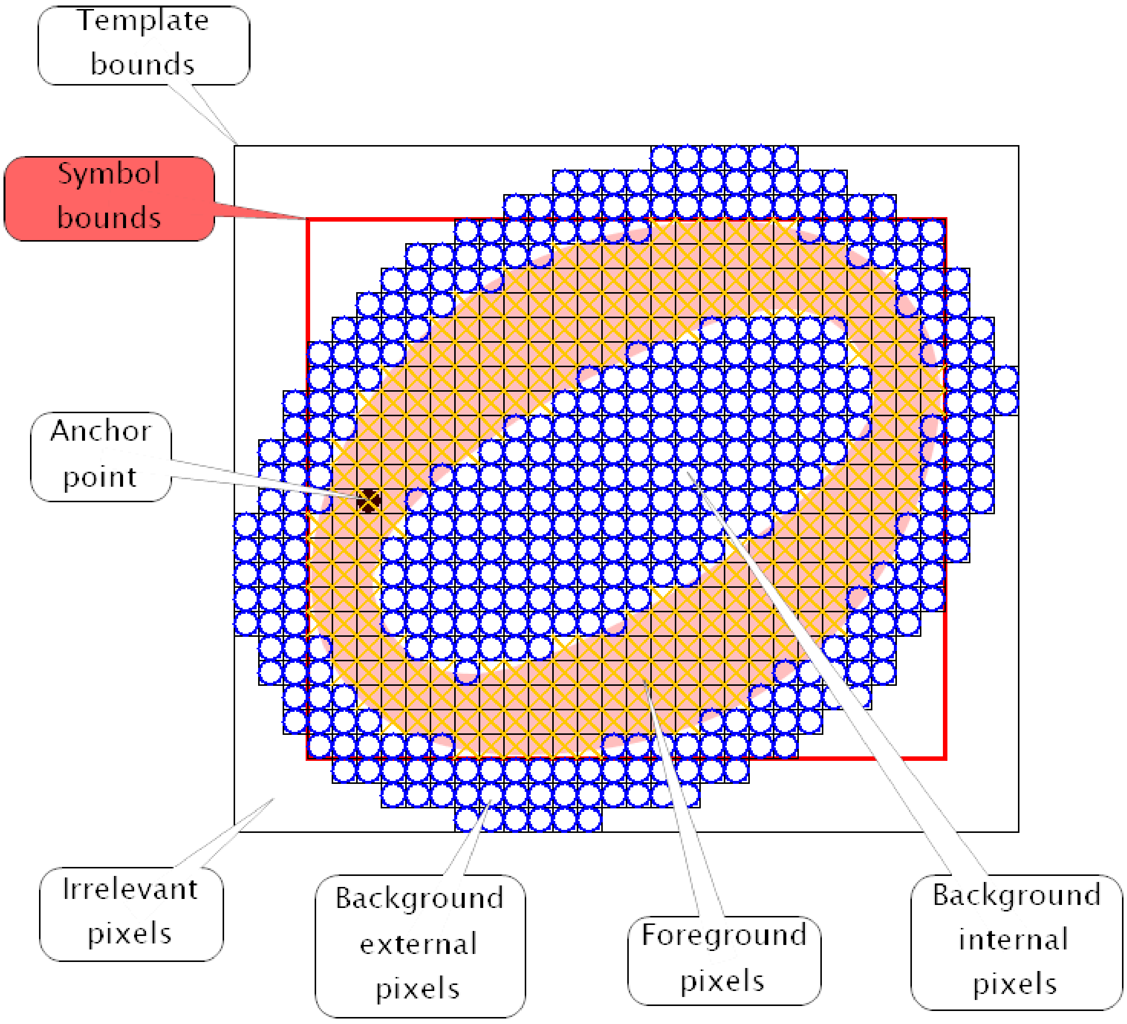
Within the shapes selection, specific processing is made for:

- Stem-based heads – they are imposed near a stem seed
- Heads with a hollow part – they are slightly boosted compared to the filled ones

Head templates

Based on the staff interline value and on the chosen music font family, a specific version of the templates is picked up from the template factory.

- For every head shape, the factory defines a template as follows (the display is here magnified 16 times)



- The above example represents the template for an oval void head:
  - It expects a stem on the lower left side of the head (hence the anchor point)
  - It defines interior and exterior background pixels (note the exterior pixels are limited to a kind of ring around the head)
  - It defines foreground pixels

Staff processing

Two passes are made along every staff, with all relevant templates:

- The first pass is driven by the stem seeds. Only abscissa locations next to a stem seed are looked up, and with just the “stem heads” templates.
- The second pass explores the entire range of possible x values, regardless of stem seeds, because of potential missing portions of stems. All heads templates are applied on every x value, so this pass is rather costly.

In both passes, every line (staff line or ledger) is scanned as well as every space between or just beyond lines (staff line or ledger).

Depending on the region within a template, a pixel match or mismatch is weighted differently:

- Weight 6 for the foreground region,
- Weight 4 for the interior background region,
- Weight 1 for the exterior background region.

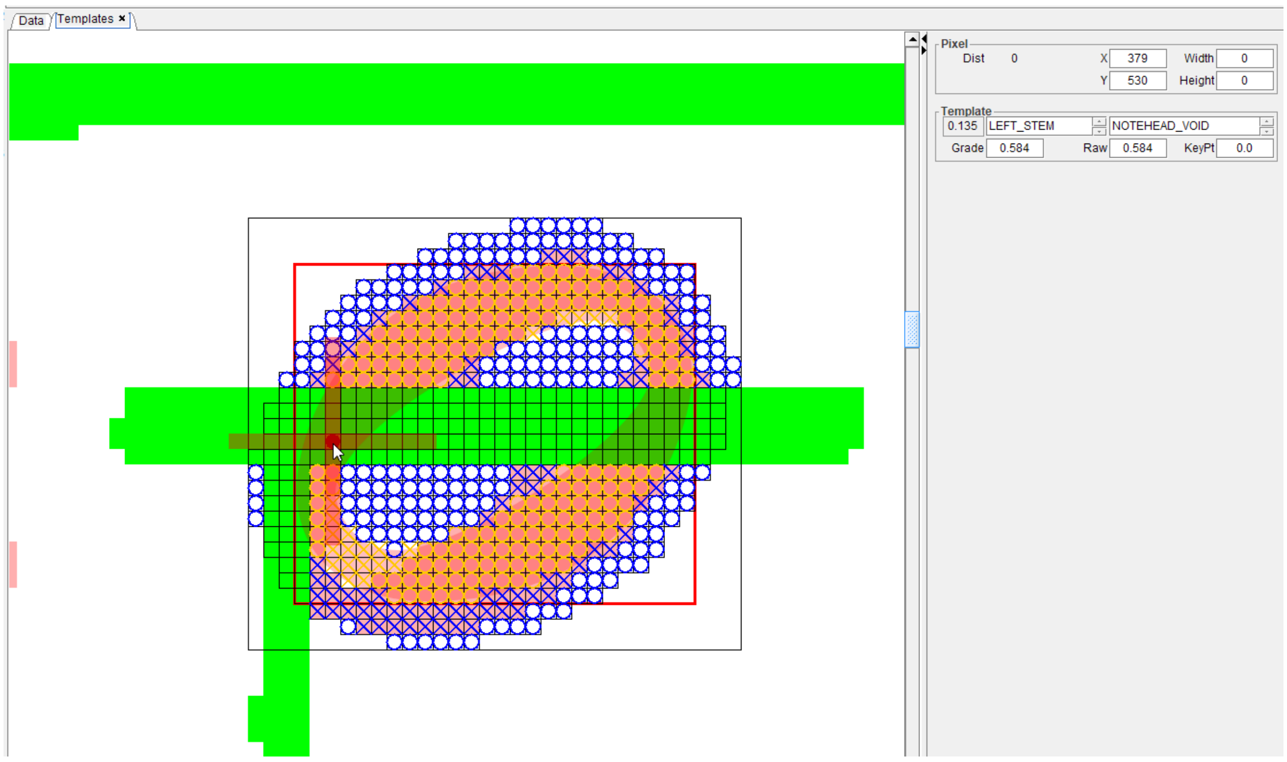
At the end of a staff processing:

- Duplicates (same shape, same bounds, same underlying glyph) are removed
- Good long beams defeat overlapping head candidates
- Good heads defeat overlapping short beams
- The remaining head candidates are inserted in the system SIG.
- Overlaps between two head candidates are detected using IOU (Intersection Over Union) criteria, and are formalized by an exclusion relationship in the system SIG.

Observing a template in action

By defaut, the `Templates` tab, which allows to interactively observe the template behavior is not presented to the end user. It requires the constant boolean `displayTemplates` to be selected in the `DistancesBuilder` class.

Note this is only a way to *study* very precisely the behavior of the template mechanism.
It is available after the `HEADS` step has been run, and is *not meant to modify its results*.



Pixel
Dist 0 X 379 Width 0
Y 530 Height 0

Template
0.135 LEFT_STEM NOTEHEAD_VOID
Grade 0.584 Raw 0.584 KeyPt 0.0

The picture above presents the application of a template at a specific location in the input image.

Note the green pixels which correspond to the points to be ignored in the input image (staff line, ledger, stem, barline).

The Template board, in the right column, allows to manually select a template by its anchor and by its shape.

The location where the template is applied is defined by the mouse right button. It is convenient to use the Alt + arrow keys to move the location one pixel at a time.

The result of applying the chosen template at the chosen location can be read in the Template board.

- The distance appears in the upper left field,
- The resulting grade in the field below.

Epilog

- The head spots, gathered by the `BEAMS` step for this `HEADS` step, can now be discarded.

- The whole population of heads candidates in the sheet are analyzed to measure the effective horizontal distance, per horizontal side and per head shape, between the head anchor and the related stem seed.
  This data is stored in the sheet Scale class, and will later be used to precisely adjust the stem reference point of every head inter.

Footnotes

[1] The distance is computed using the Chamfer Distance algorithm with a 3x3 kernel ↩

STEMS step

DOCUMENTATION NOT YET PROVIDED

REDUCTION step

DOCUMENTATION NOT YET PROVIDED

CUE_BEAMS step

DOCUMENTATION NOT YET PROVIDED

TEXTS step

DOCUMENTATION NOT YET PROVIDED

MEASURES step

DOCUMENTATION NOT YET PROVIDED

CHORDS step

DOCUMENTATION NOT YET PROVIDED

CURVES step

DOCUMENTATION NOT YET PROVIDED

SYMBOLS step

DOCUMENTATION NOT YET PROVIDED

LINKS step

DOCUMENTATION NOT YET PROVIDED

Explanation / Steps internals / RHYTHMS step

RHYTHMS step

DOCUMENTATION NOT YET PROVIDED

PAGE step

DOCUMENTATION NOT YET PROVIDED