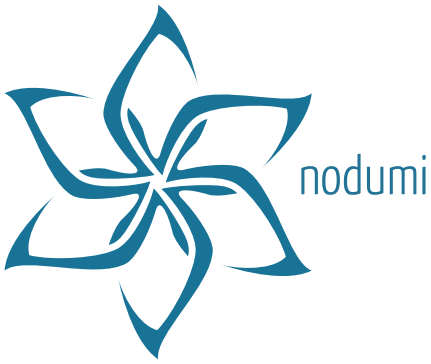
User Manual

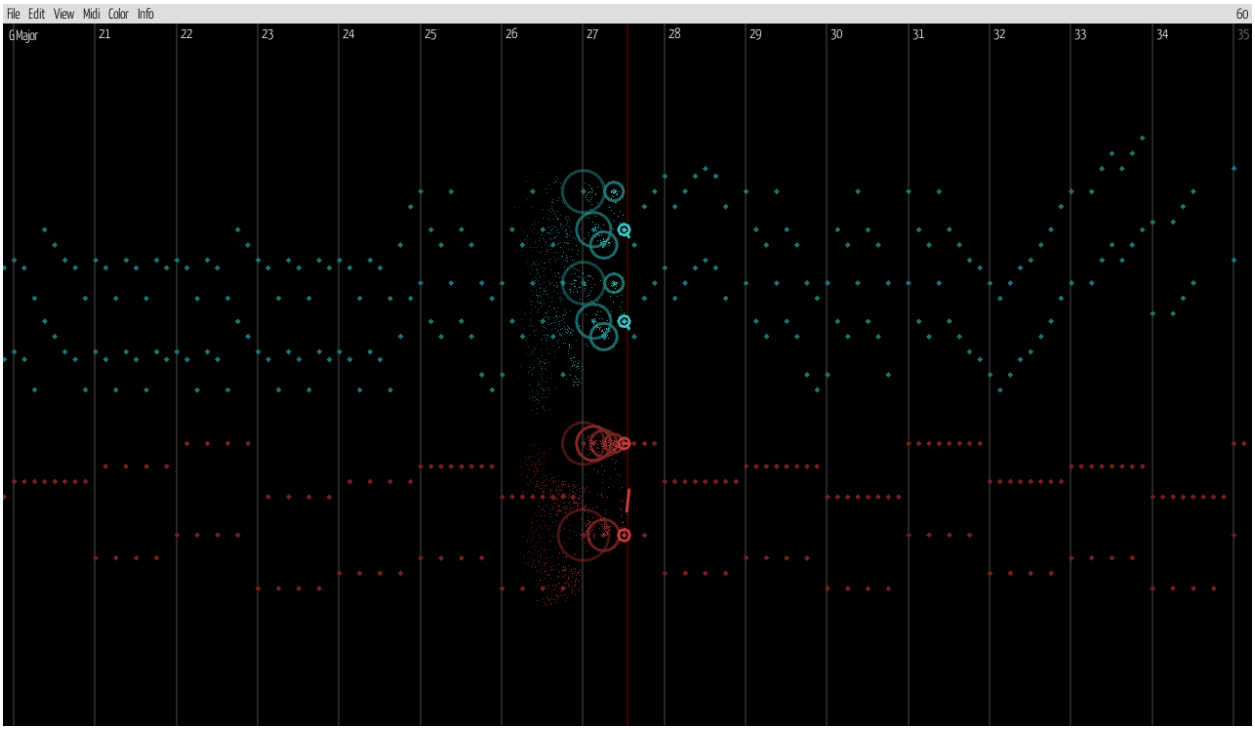
`nodumi` – Interactive MIDI Visualizer

24 January 2024



Contents

1 Introduction



`nodumi` is an interactive MIDI visualizer. Both live input (such as from a MIDI-enabled instrument) and prerecorded MIDI files are supported. Moreover, this program defines, implements, and supports a custom file format (MKI) for parametrizing visualization schemes. `nodumi` is inspired by Stephen Malinowski's *Music Animation Machine* (MAM2006), and aims to function as a modernized version of the public-facing MAM2006 program. Many features from MAM2006 are present, and additional features have been added as well. Notably, support for both Windows and many Linux systems is provided, in contrast to the Windows-exclusive nature of MAM2006. Currently, macOS is not supported due to the specific resources required to create builds for the Mac ecosystem. This software is licensed under GPLv3 – therefore the source code is made freely available: (iikare/nodumi).

This application was created by iika-re.

2 System Requirements

For Linux systems, you will need to install `raylib`, either through your distribution's package manager or by directly building and installing the required shared libraries. The `openmp` package is also a requirement for compilation. You will also need the libraries output by the command `pkg-config --libs gtk+-3.0`. As of 13 October 2023, these are (identified via corresponding linker flag):

```
-lgtk-3 -lgdk-3 -lz -lpangocairo-1.0 -lpango-1.0 -lharfbuzz -latk-1.0 -lcairo-gobject
            -lcairo -lgdk_pixbuf-2.0 -lgio-2.0 -lgobject-2.0 -lglib-2.0
```

For Windows systems, all dependencies are bundled into the executable. A reasonably modern OS is required (testing has only been done on Windows 10, but theoretically earlier Windows versions should be compatible).

A GPU supporting OpenGL 3.3 or higher is required. A relatively modern CPU is desirable, but not necessary. The performance demands of this application are low unless you wish to visualize very large MIDI files. One exception is the Voronoi rendering mode, which requires a reasonably performant GPU to maintain an acceptable framerate.

3 Obtaining Binaries

This section entails how to obtain nodumi – either downloading prebuilt binaries for supported operating systems, or by building directly from source.

3.1 Prebuilt Binaries

At the moment, there is no set release schedule. Prebuilt binaries are often (but not always) available at (iikare/nodumi) in the bin directory. Use nodumi for Linux-based systems, and nodumi.exe for Windows-based systems. Note that there is *no* installer for either operating system. As there are no persistent configuration files, the application is portable by default.

3.2 Compilation from Source

NOTE – building from source requires a Linux development environment, as the compilation process depends on tools that are not Windows-compatible. This means that both Linux and Windows executables must be built from Linux.

First, obtain the source code by cloning (iikare/nodumi). Then, run `make rel=y` for a Linux release build, or `make arch=win rel=y` for a Windows release build. Compile-time dependencies are listed as submodules in the directory, so make sure to obtain them as well. Cross-compilation from Linux to Windows requires `x86_64-w64-mingw32-g++`, `x86_64-w64-mingw32-gcc`, and `x86_64-w64-mingw32-ld`. The final executable can be then found in the `bin` directory.

4 Usage

A very brief usage overview is provided below. Further usage information is given in a later section.

4.1 Command Line Interface

While there is no true text-based interface, both a MIDI/MKI file and an image can be loaded automatically through the proper command line arguments. Any order of arguments is accepted, with the first MIDI/MKI file or image file in the list being taken. Any MIDI/MKI or image files past the first of each type and any invalid paths are ignored. An example is given below.

```
./bin/nodumi [path to MIDI/MKI] [path to image]
```

There are no other command line parameters. After executing the above command, the graphical interface will start automatically.

4.2 Graphical Interface

If not starting the application from the command line, the graphical interface will open automatically without any files loaded.

4.3 Warnings and Errors

Almost all warnings and errors are reported at the command line. Therefore, you will need to view the terminal output to diagnose any potential problems. Log files are not automatically generated on each program run. If you want a method of generating this log file, pipe `stdout` to a temporary log file. This will require you to launch `nodumi` from the command line. An example is shown below.

```
./bin/nodumi [args] > "nodumi_`date +%Y%m%d%H%M%S`.log"
```

5 Keyboard Shortcuts

For ease of use, keyboard shortcuts have been implemented for common actions. They are able to be used both through conventional means as well as through a set of vim-like bindings. There is currently no option to rebind shortcuts (of either type).

5.1 Standard

A set of shortcuts (mostly) conforming to conventional norms are defined via the table below.

Key	Action
Ctrl-O	Open MIDI/MKI File
Ctrl-Shift-O	Open Image File
Ctrl-W	Close MIDI/MKI File
Ctrl-Shift-W	Close Image File
Ctrl-S	Save (to MKI)
Ctrl-Shift-S	Save As (to MKI)
Ctrl-R	Reload MIDI/MKI File
Ctrl-,	Open Preferences
Ctrl-F	Open File Info
Ctrl-I	Open Program Info
F7	Exit
Ctrl-Space	Toggle Live Play Mode
Space	Toggle Playback
Ctrl-$n, n \in [1, 9]$	Enable n^{th} Display Mode
Ctrl-Home	Jump to Start
Ctrl-End	Jump to End
Scr.Wheel	Zoom
Alt-Scr.Wheel	Zoom (precise)
$\uparrow \cdot \downarrow$	Zoom
$\leftarrow \cdot \rightarrow$	Move Back/Forward
Ctrl-$\leftarrow \cdot \rightarrow$	Move Back/Forward (faster)
Shift-$\leftarrow \cdot \rightarrow$	Move Back/Forward (by measure)
Ctrl-Scr.Wheel	Image Zoom
Shift-Scr.Wheel	Image Zoom (precise)

5.2 vim-like

Similar to the list in Section 5.1, these shortcuts are bound to common actions. These use the command buffer, which is visible at the bottom right corner of the window. Pressing `esc` or enter will clear the buffer. Once a valid command is input to the buffer, it will either be (a) executed immediately if not beginning with `:`, or (b) executed after `<Enter>` is pressed by the user. The list of these bindings is presented below.

Binding	Action
`:o`	Open MIDI/MKI File
`:oi`	Open Image File
`:q`	Close MIDI/MKI File
`:qi`	Close Image File
`:r`	Reload MIDI/MKI File
`:qa`	Exit
`:p`	Open Preferences
`:f`	Open File Info
`:i`	Open Program Info
`:l`	Toggle Live Play Mode
`:s`	Toggle Sheet Music Display
`:d`$n, n \in [1, 99]$	Enable n^{th} Display Mode
`gg`	Jump to Start
`G`	Jump to End
n`G`	Jump to n^{th} Measure
`b`	Move Back One Measure
n`b`	Move Back n Measures
`w`	Move Forward One Measure
n`w`	Move Forward n Measures

6 Features

An explanation of the major features of this program follows.

6.1 Display Modes

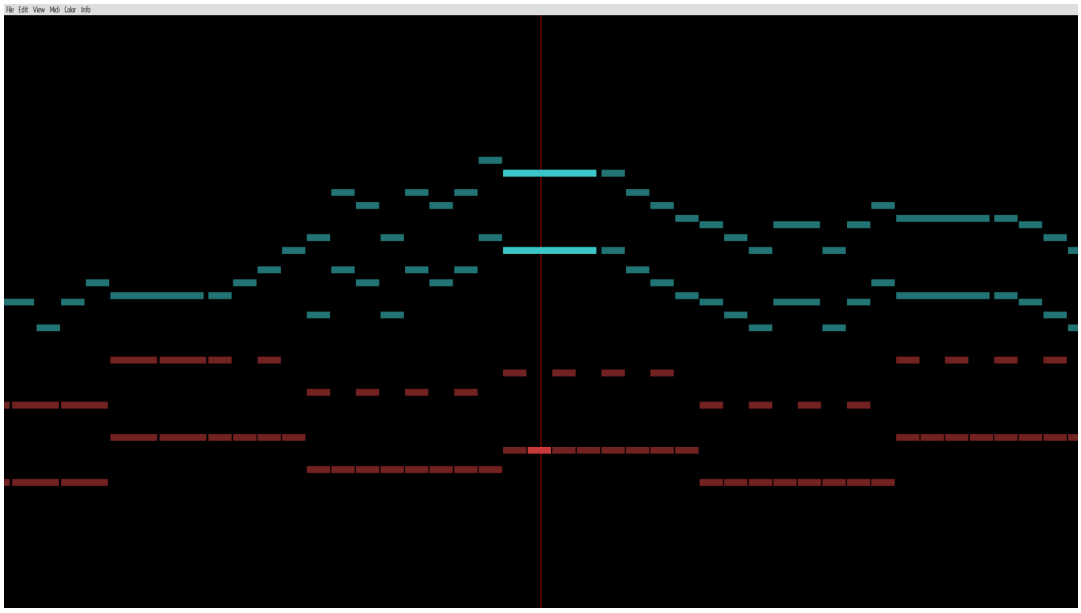
Currently, there are six fully implemented display modes, all of which are usable for both live input and while using prerecorded files. For all display modes, the x direction represents time, and the y direction represents the 88 keys playable on a standard piano (i.e. pitch). While the MIDI specification provides for note input in the key range 0-127, only values 21-108 are visualized by this application. There are three classes of display modes:

Bar-based	Ignores temporal context of notes.
Line-based	Accounts for temporal context of notes.
Other	Not encapsulated by the above types.

Each mode is marked with its corresponding type. Line-based modes link temporally adjacent notes on the same track.

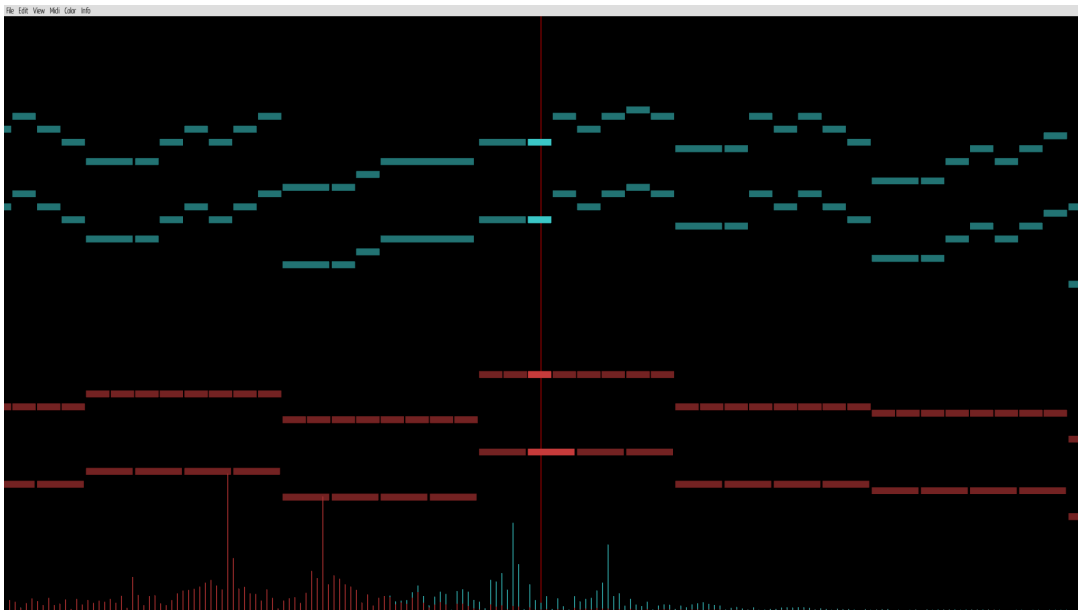
All display modes are accessible via `View` → `Display Mode:` → `[mode]`.

6.1.1 Bar



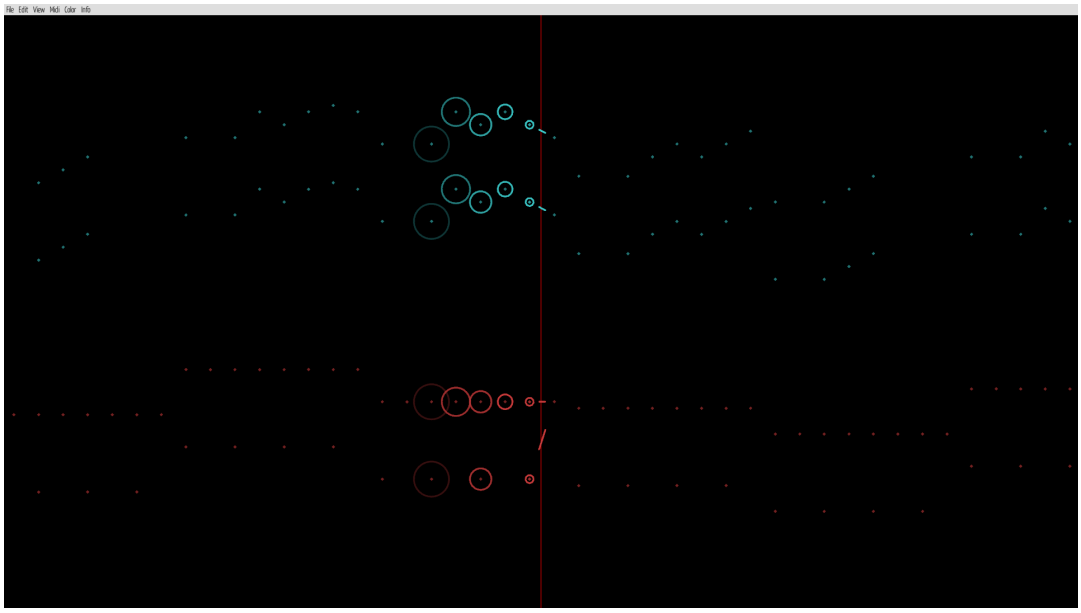
The simplest display mode – this simply draws a bar at the x position of the note with length determined by its duration. This is a BAR-BASED display mode.

6.1.2 FFT



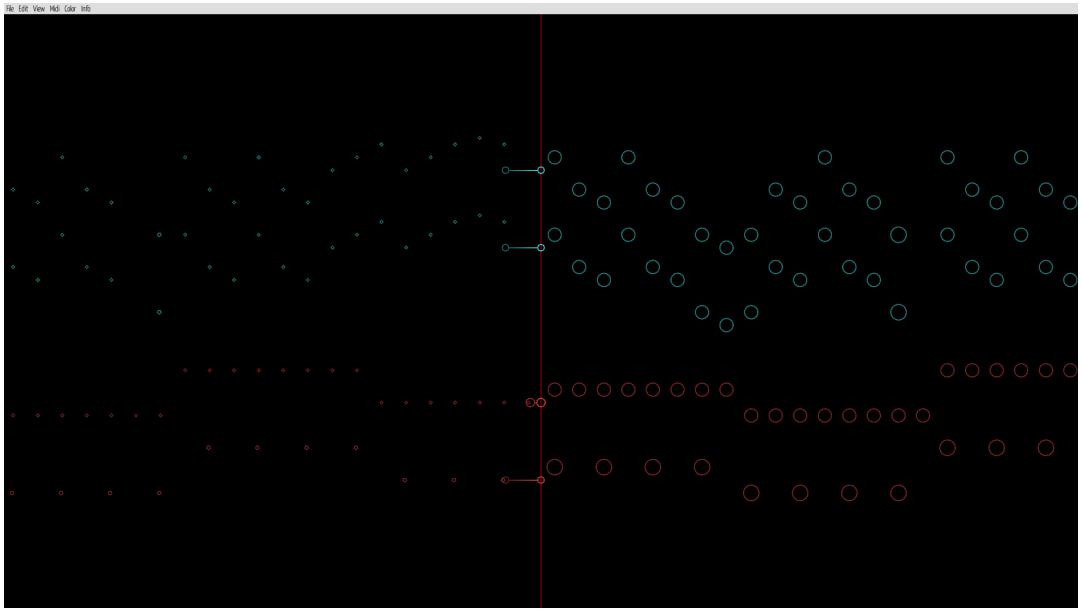
Identical to the BAR mode, with the addition of note frequencies via an FFT approximation algorithm. The amplitude of a given frequency bin is displayed along the bottom of the screen, and all bins combined encompass the frequency range $[20, 44100]$Hz. Since MIDI data does not contain actual samples, an envelope function must be chosen to approximate bin lengths. Currently, this function is only modelled for a piano even when another MIDI instrument is present on track. Amplitudes are calculated for each track, and then a summation over all tracks is done to obtain the final frequency bin distribution. This can be seen in the overlapping colors in the spectrum about halfway across the above image. In practice, when accounting for overtones, harmonics, and other signal behavior, not all of the spectrum is used equally (note that C8 on a piano only emits 4.2KHz). The display is logarthimic in frequency. This is a BAR-BASED display mode.

6.1.3 Pulse



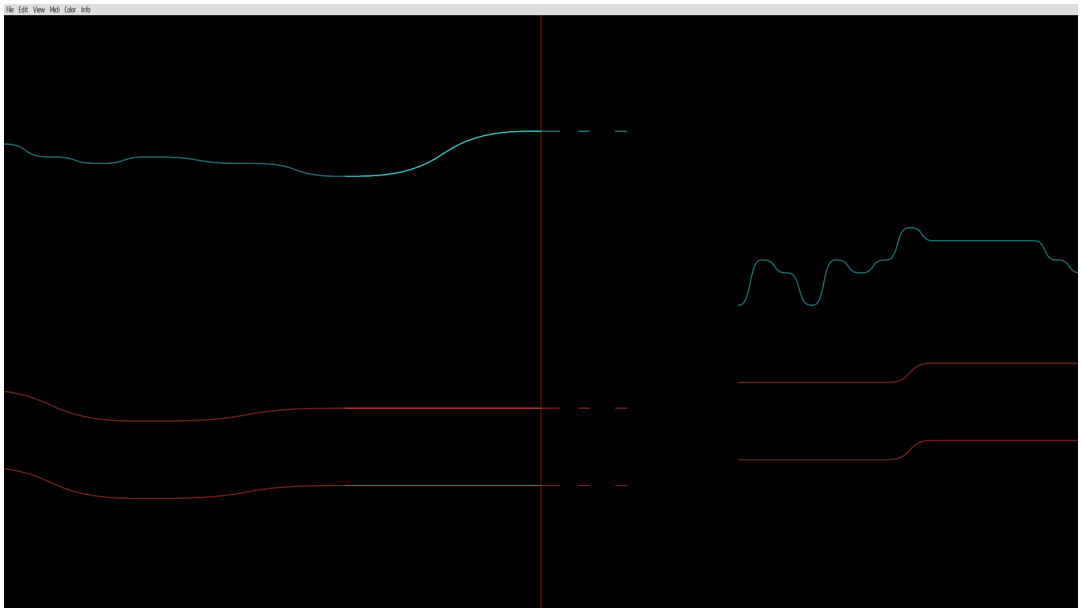
A portion of the straight line between points is drawn instead. The length of this line segment is dependent on the relative position and duration of the note in respect to the *now*-line. Additionally, a pulse appears at a note-on event and fades out over time. This is a LINE-BASED display mode.

6.1.4 Ball



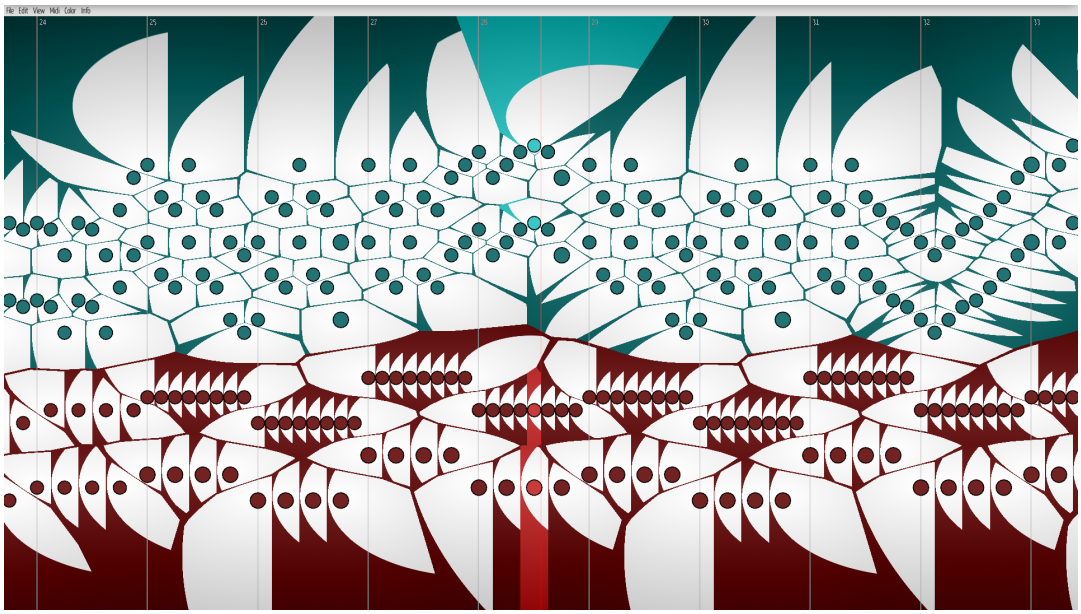
A large circle slowly shrinks as a note is played, and follows the *now*-line. Along with it, a circle is also drawn at the note's original start position, and a line connects the two circles. Once the duration of the note has passed, only a small circle remains at that note's ending position. This is a BAR-BASED display mode.

6.1.5 Line



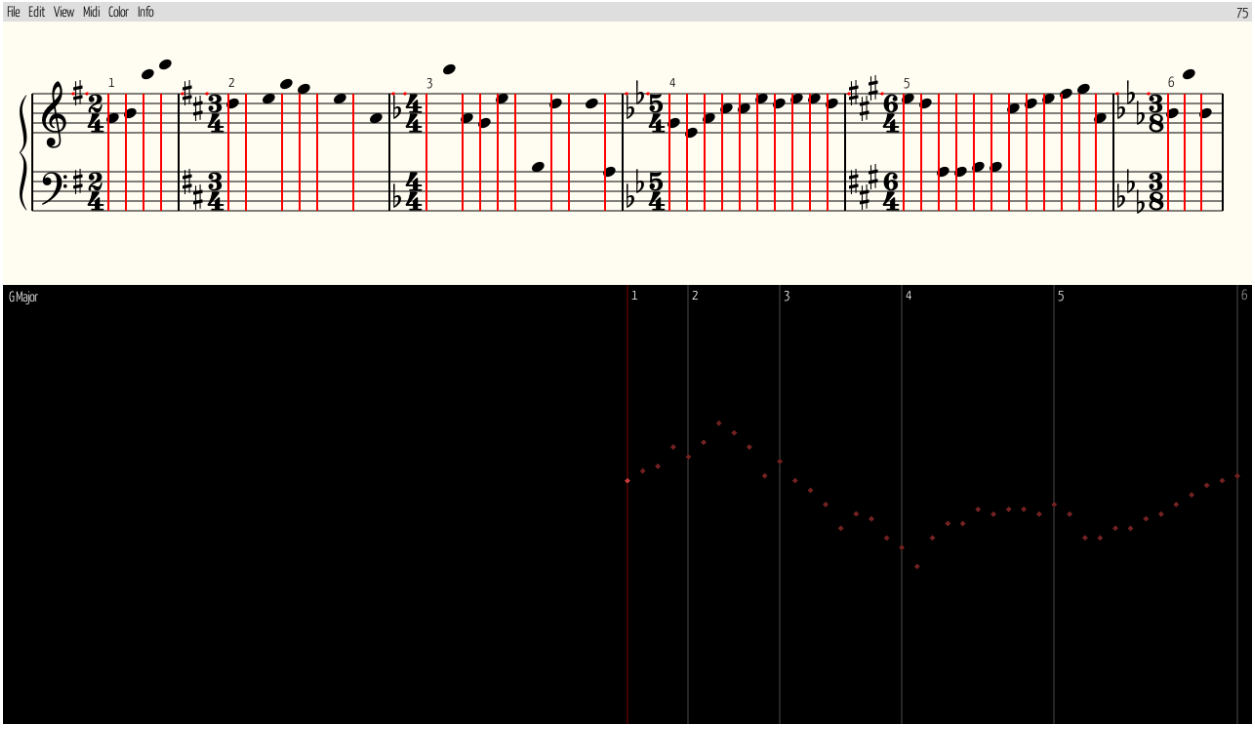
This mode draws a continous bezier curve between notes. For line segments shorter than a given threshold, a single straight line segment is drawn to optimize performance. This is a LINE-BASED display mode.

6.1.6 Voronoi



Spatial distance is encoded in a Voronoi-based diagram, calculated in real-time via GPU. Additionally, a circle is placed at the start possition of each note to mark the origin of each Voronoi cell. The algorithm used to create this visualization is $O(k^2)$ in the number of points, so performance may suffer on older, slower, or integrated GPUs. Image backgrounds cannot be seen when using this visualization. This is an OTHER display mode.

6.2 Sheet Music Generation



The ability to generate sheet music from MIDI/MKI files or from live input is supported. Keep in mind this feature is still in development, and the current progress on this feature is not representative of its final state.

Take note of the ill-placed note markings and lack of flags or beaming. Future iterations of the sheet music creation algorithm are planned to fix these shortcomings. It is best to think of the sheet music generator as a technical preview rather than a full-fledged feature.

This feature is accessible via `Edit` → `Enable Sheet Music`.

6.3 Song Time

This option places a marker of the current piece's position in the upper right corner of the display. Two options are provided for the format: *Relative Song Time*, and *Absolute Song Time*. For live input, this value is proportional to the amount of time spent in the Live Play mode.

6.3.1 Relative

The *Relative Song Time* option displays a percentage that corresponds to the percentage of the piece complete relative to the *Now*-Line. For live input, this value is not rendered at all instead.

This feature is accessible via `View` → `{Display, Hide} Song Time` → `Relative`.

6.3.2 Absolute

The *Absolute Song Time* option displays two numbers – the first one corresponds to the current amount of time elapsed for the current file, while the second is the total length of the file. For live input, this value consists of only one number – the amount of time since the Live Play mode was initiated.

This feature is accessible via `View` → `{Display, Hide} Song Time` → `Absolute`.

6.4 Key Signature Detection

For MIDI/MKI files, an option to extract and render the current key signature from the file is provided. In most cases, key signature data is generated directly from the MIDI/MKI file. If an input MIDI file does not have any key signatures, the key signature with the smallest number of off-key notes is used instead. Currently, this assumes the entire piece uses the same key signature. For live input, a heuristic-based key detection algorithm is planned.

This feature is accessible via `View` → `{Show, Hide} Key Signature`.

6.5 Tempo Detection

For MIDI/MKI files, an option to extract and render the current tempo from the file is provided. For live input, this feature is not available.

This feature is accessible via `View` → `{Show, Hide} Tempo`.

6.6 *Now*-Line

A vertical line is displayed at the x-position of the current time marker. This line is static in relation to the notes. It is possible to reposition the line to any valid x-coordinate. To do this, left-click on the line and drag to the intended position. When this is done, the notes will move along with the line as well. In addition, to reset the position of the line, hold `Shift` while left-clicking.

This feature is accessible via `View` → `{Display, Hide} Now Line`.

6.7 Measure Lines

Like the *Now*-Line, vertical lines are placed at the start of each measure. These lines cannot be moved, as they are fixed to the timestamp at which they appear in the MIDI/MKI file. There are no measure lines displayed when the Live Input mode is active.

This feature is accessible via `View` → `{Display, Hide} Measure Line`.

6.8 Measure Numbers

With this option, each measure has a corresponding measure number displayed at the top of the screen. The number is rendered slightly to the right of the measure line to prevent overlap. When zooming out, only a subset of measure numbers are shown in order to preserve readability.

This feature is accessible via `View` → `{Display, Hide} Measure Number`.

6.9 Hide Background

If an image is loaded, this option prevents it from being rendered. Otherwise, this option has no effect.

This feature is accessible via `View` → `{Show, Hide} Background`.

7 Real-Time I/O

Communication to external MIDI devices is possible. This includes sending MIDI events to a synthesizer, and processing incoming MIDI data from a device. This option also allows the user to play a MIDI instrument and view a live visualization at the same time. When MIDI ports are displayed, a short version of the device-provided port name is provided, and each input or output port is uniquely numbered.

7.1 Input

If an input MIDI port is selected, the program will listen on that port for incoming MIDI messages. Valid messages will be displayed on screen as if they came from a prerecorded file. Note that an input port can be opened even when the Live Play mode is not activated.

This feature is accessible via `Midi` → `Input` → `[port]`.

7.2 Output

If an output MIDI port is selected, the program will output MIDI messages to that port. If a file is loaded, sent messages correspond with the currently playing notes. If the Live Input mode is selected (and a valid input port is open), the program will instead act in a MIDI Through regime. Therefore, input notes will still be rendered while allowing for audio to be provided by an external synthesizer.

NOTE: Using virtual MIDI ports on Windows is not supported, and will likely crash the application. This has been reproduced by creating virtual ports with `loopMIDI` and trying to send MIDI messages via those ports.

This feature is accessible via `Midi` → `Output` → `[port]`.

7.3 Live Play Mode

This mode allows live MIDI data to be displayed on screen. To do this, connect a MIDI device to the computer, enable this mode, and then select the corresponding MIDI input port via `Midi` → `Input` → `[port]`.

This feature is accessible via `Midi` → `{Enable, Disable} Live Play`.

8 Color Selection

Almost every UI element has a user-controllable color parameter. This section details the ways in which these parameters can be manipulated by the user.

8.1 Color By

This function allows for the coloring of notes via different methods. Each note has two associated colors at all times – one for when it currently playing or selected by the user, and another for all other cases. These options allow setting both of these colors for all notes at the same time using a predefined algorithm. Even when using these options, the user can still manually override any algorithmically generated color.

Object	Controllable Values
Note	{On, Off} Color (for each {Track, Tonic Offset, Velocity Value})
Now-Line	Line Color
Measure Lines	Line Color
Sheet Music	Background Color
Background	Background Color

8.1.1 Part

In this coloring mode, each track has a unique on/off color. In addition, for MIDI/MKI files, the hue of the generated color is determined by ordering each track by its average y value (i.e. the MIDI note #), and assigning a color based on the track's position in this ordering.

This feature is accessible via `Color` → `Color By:` → `Part`.

8.1.2 Velocity

MIDI events provide for each note a velocity value $V \in [0, 127]$. Then, the color is determined by simply mapping the note's velocity to a set of 128 possible colors. This set of colors is evenly-distributed around the HSV color wheel, and each color only differs in hue.

This feature is accessible via `Color` → `Color By:` → `Velocity`.

8.1.3 Tonic

This colors notes based on the musical concept of *tonic*. The tonic can be set manually via right clicking on a a note and pressing the `Set Tonic` option. This will set the reference tonic as the selected note. Note that you can set the tonic even while this option is not enabled. In that case, you need to enable this option to see your changes.

This feature is accessible via `Color` → `Color By:` → `Tonic`.

8.2 Color Scheme

A color scheme is able to be derived from the provided background image. By default this is disabled (i.e. this option is preset to the `Default` mode). When an image is loaded, the `From Background` mode is available.

8.2.1 Default

The default color scheme. If an image is loaded, it is not used to calculate any element color values.

This feature is accessible via `Color` → `Color Scheme:` → `Default`.

8.2.2 From Background

The note color scheme is determined using the background image. This uses a randomized *k-means* algorithm to pick the amount of colors. Note that the random nature of this process sometimes leads to unsatisfactory results. In that case, simply rerun the operation until the desired color palette has been generated. This operation will generate values for every *Color By* option simultaneously. Using this operation on an image with less than $\max(\#\text{ Tracks}, 12)$ colors is undefined behavior, and is therefore not supported. In this case, the program will emit a warning.

This feature is accessible via `Color` → `Color Scheme:` → `From Background`.

8.3 Color Swapping

This option swaps the note-on and note-off color sets without changing any other color options.

This feature is accessible via `Color` → `Swap Colors`.

8.4 Color Inversion

This option inverts the object colors specified by the table below. For any given RGB input color $C = (C_r, C_g, C_b)$ such that $C_r, C_g, C_b \in [0, 255]$, the inverted color C' is defined as $C' = (255 - C_r, 255 - C_g, 255 - C_b)$.

This feature is accessible via `Color` → `Invert Color Scheme`.

Object	**Invertible Values**
Note	{On, Off} Color (for each {Track, Tonic Offset, Velocity Value})
Now-Line	Line Color
Measure Lines	Line Color
Sheet Music	Background Color
Background	Background Color

9 The MKI File Format

Normal MIDI files do not provide the ability to store visualization-related metadata such as color scheme, rendering options, etc. Therefore, the MKI file format has been created to alleviate this problem. This program defines, implements, and provides support for the MKI file format. Notably, every visualization-related parameter accepted by this program is able to be stored in the MKI format. A MIDI bitstream is embedded into the format, and user-provided images are able to be stored as well.

To save a file in MKI format, use the `Ctrl-S`/`Ctrl-Shift-S` keyboard shortcuts, which correspond to the `File` → `Save`/`File` → `Save As` options.

9.1 Specification

The MKI file format is open and non-proprietary – anyone is free to use the specification described below. However, due to the frequency changing nature of this program, this specification is subject to change at any time. This software provides no guarantees that a MKI file created today will be compatible with future versions of the program. Moreover, many of the options are very specific to this program.

The MKI format is defined as follows:

Offset[start:end]	**Value**
`0x00[7:7]`	*Now*-Line
`0x00[6:6]`	Display FPS
`0x00[5:5]`	Display Background
`0x00[4:4]`	Display Sheet Music
`0x00[3:3]`	Display Measure Line
`0x00[2:2]`	Display Measure Number
`0x00[1:1]`	Image Exists?
`0x00[0:0]-0x02`	`[reserved]`
`0x03[7:4]`	Color Scheme
`0x03[3:0]`	Display Mode
`0x04[7:4]`	Song Time Mode
`0x04[3:0]`	Tonic Offset
`0x05-0x07`	`[reserved]`
`0x07-0x0B`	Zoom
`0x0C-0x1F`	Image Metadata (if exists)
`0x20-0x67`	Tonic Color Data
`0x68-0x367`	Velocity Color Data
`0x368-0x36A`	Background Color
`0x36B-0x36E`	Track Size Marker
`0x36F-0x36F+N`	Track Color Data
`0x36F+N-0x36F+N+M`	Raw MIDI Data
`0x36F+N+M-0x36F+N+M+P`	Raw Image Data (if exists)

N refers to # Tracks $* 6$, M is the size of the MIDI bitstream, P is the size of the image bitstream.

10 User Preferences

This section pertains to the file-agnostic preferences a user can set. All of these options are available via the keyboard shortcut `Ctrl-,`, which corresponds to the `Edit` → `Preferences` action.

10.1 Particle Emission

This option controls the particle system. Depending on the selected display mode, particles will be emitted from a predefined location on a note-on event. In all cases, the location of the particle emission is related to the location of the displayed note. A pictorial example is provided in the introduction.

10.2 Dynamic Label Coloring

The label color for measure numbers, key signatures, etc. may not be visible on bright backgrounds. Therefore, this option exists to automatically select the grayscale RGB color $C_{\text{RGB}} = (c, c, c), c \in [0, 255]$ given the background color B_{RGB}, that maximizes $\Delta E(C_{\text{LAB}}, B_{\text{LAB}})$. This color will then be set as the default label color.

10.3 Adaptive Track Division

Many MIDI devices such as digital pianos only support single-channel MIDI output. Then, it can become difficult to discern the hand that plays a specific section of a piece. This option enables an algorithm that processes live MIDI input to split single-channel input into two tracks automatically. Options such as the time differential between successive notes, player hand size, and other factors are taken into account. Note that the algorithm output is tuned for use with MIDI-enabled pianos, so using other two-handed (or greater) instruments as input may lead to confusing results.

Note that it is possible to automatically divide one-channel tracks as well. An option for this is present in the user preferences screen. There are separate options for track division – one for live input, and one for MIDI/MKI files. If the corresponding option for MIDI/MKI files is enabled after a single-track MIDI/MKI file is loaded, the user must reload that file for this option to take effect. This option has no effect on multi-track MIDI/MKI files.

10.4 Hand Range Selection

This option is available only when the Adaptive Track Division option is enabled. The user is able to select the maximum hand range (8^{th}, 9^{th}, 10^{th}, or 11^{th}) that suits them best. The track division algorithm will then take the user's choice into effect. The default hand range is a 10^{th}.

10.5 Image Adjustment

If the user-supplied image is too bright, a darkening option has been provided. Note that full image manipulation is beyond the scope of this program, and so if the user has a problem that cannot be solved via this option, they should fix their issue by using an external image editor.

10.6 Color Distance Calculation

This program supports ΔE calculation via three different color distance algorithms: CIE76, CIE94 and CIEDE2000. The user is able to choose freely between these algorithms, which will then be used throughout the entire program. The default color distance algorithm is CIEDE2000. A table summarizing the benefits of each follows.

ΔE Formula	**Properties**
CIE76	Fastest, Least Accurate
CIE94	Moderate Speed & Accuracy
CIEDE2000	Slowest, Most Accurate

10.7 Display FPS

It simply displays the current FPS at the top right corner.

This feature is accessible via `View` → `{Show, Hide} FPS`.

10.8 Velocity Scaling

The default velocity color mode expects velocities in $[0, 127]$. Therefore, if a piece uses a small subset of this range, the contrast between note colors will be low. This option allows the user to map the full color range provided by this color mode to the notes in this file. Currently, this option only works for prerecorded pieces, not live input. If a piece has a velocity range of zero, then this option does nothing and the default velocity coloring will be shown.

10.9 Shadow Effects

A drop shadow effect is available to the user. Currently the angle of the shadow is fixed at 45°. The shadow distance is adjustable by the user via the built-in slider.

11 Frequently Asked Questions

Q – How can I enable audio output?

A – This program does not itself produce audio. This is by the nature of MIDI messages, which do not encode audio samples, but rather note data & metadata. You can enable the MIDI output feature by following the instructions in the Output section, which will allow you to hear note events with an external synthesizer.

Q – I enabled MIDI Input, but there is no visualization shown. How can I fix this?

A – Assuming your MIDI device is working correctly, the most likely explanation is that you have not activated the Live Play Mode.

Q – I enabled MIDI Output, but my audio quality is subpar. How can I fix this?

A – The quality of software MIDI synthesizers greatly varies. Often, a simple way to make a MIDI synthesizer sound far better is to install a custom soundfont. A good free option is `FluidR3_GM.sf2`, available in many package repositories. For Windows, if you are stuck using the terrible `Microsoft GS Wavetable Synth` as your MIDI output device, finding a better alternative is strongly recommended.

Q – Why do some visualizations look different in the Live Play Mode?

A – These visualization modes expect "fully-formed" notes. Since the duration of notes played via live MIDI input cannot be inferred until the note ends, strange behavior often occurs. Even then, all visualization modes still function – albeit to a lesser standard of quality – during live input.

Q – Are languages other than English supported?

A – While support for changing languages exists, there are currently no translations available. Therefore, English is the only supported language. Community translations are welcome, and translation is possible simply by duplicating `LABEL_LANG_EN` in `src/data_str.h`, replacing `EN` with the respective ISO 639-1 language code, and replacing the *value* of each map entry with the correct translated string. Note that the keys *must* be kept in English – they are internal identifiers and are required for the program to function. However, these internal identifiers will not be shown to the end-user.

Q – How is this software licensed and is it free?

A – This program is provided free-of-charge to everyone. Instructions to download the software are provided in the Obtaining Binaries section. The license chosen is the GNU General Public License v3.0 (GPLv3), and the license terms are provided in the repository. As a summary, the end user is guaranteed the freedom to run, study, share, and modify this program subject to the stipulations in the GPLv3 license.

12 Remarks

This program is open-source (iikare/nodumi), and community contributions are welcome. Additional information can be found on my website.