

The `mousik` package*

Celia Rubio Madrigal†

August 24, 2026

Abstract

`mousik` is a package for writing music in L^AT_EX. It is based on TikZ and includes staves, clefs, key and time signatures, notes and chords, rests, beams and tuplets, accidentals, articulations, slurs and ties, dynamics and hairpins, and piano notation. It uses short commands and allows the user to control the position and shape of the musical elements. Everything is written directly in L^AT_EX without an external music preprocessor.¹

Keywords

mousik, music, music theory, music notation, music writing, score, sheet, tikz diagram, bar, clef, time signature, note, symbol, rest, dot, chord, accidental, sharp, flat, natural, key

Contents

1	Introduction	2	3.7	Other symbols	9
2	The <code>mousik</code> environment	2	3.7.1	<code>\Dotted</code>	9
3	Available commands	3	3.7.2	<code>\Tie</code>	9
3.1	<code>\Barline</code>	3	3.7.3	<code>\Slur</code>	10
3.2	<code>\Space</code>	4	3.7.4	The <code>Slurred</code> environment . . .	10
3.2.1	<code>\BreakLine</code> . .	4	3.7.5	The <code>Crescendo</code> and <code>Decrescendo</code> environments . .	10
3.3	<code>\Clef</code>	4	3.7.6	<code>\Artic</code> and <code>\Symbol</code>	11
3.4	<code>\TimeSig</code>	5	3.7.7	<code>\Appog</code>	12
3.5	Notes and rests	5	3.7.8	<code>\Text</code> and <code>\Dynamics</code> . . .	13
3.5.1	<code>\Note</code>	5	3.7.9	<code>\Tempo</code>	13
3.5.2	<code>\Rest</code>	7	4	Tips and observations	13
3.6	Keys and accidentals . .	7	5	Notes from the author	15
3.6.1	<code>\Acc</code>	7			
3.6.2	<code>\Key</code>	8			
3.6.3	<code>\NoKey</code>	8			
3.6.4	<code>\MixedKey</code> . . .	8			

*This document corresponds to `mousik` v0.1, dated 2026/08/24.

†Email: rubiomadrigalcelia@gmail.com

¹The code is also hosted at <https://github.com/celrm/mousik>.

1 Introduction

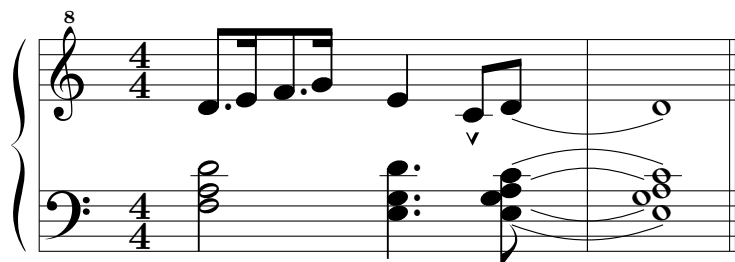
The `mousik` package renders music scores with $\text{\LaTeX}$. It is based on the `tikz` package, and the scores are treated as drawings. The package does not try to understand the music, and it does not decide how to lay out the musical elements. It provides short commands for notes, rests, beams, slurs, accidentals, articulations, etc., and lets the user change the position and shape of these elements directly. This makes it most useful for small examples.

There are several other ways to write music with $\text{\LaTeX}$, but they generally require different programs, notations, or processing steps. `LilyPond` has its own language and program, and you may need extra tools such as `lilypond-book` or `LyLuaTeX` to include it in $\text{\LaTeX}$. As for the `abc` package, the music is written in ABC notation and converted by an external program. `MusiXTeX` works directly with $\text{\TeX}$, but it requires very detailed score descriptions, and it also uses a second pass to calculate spacing. `PMX` makes `MusiXTeX` easier to write, but it uses its own notation and a conversion step.

In comparison, `mousik` is meant to be simple, and is designed to avoid an external notation or preprocessing step.

2 The `mousik` environment

A score is drawn by means of the `mousik` environment.



```
\begin{mousik}[piano, piano-sep=2]
  \UpperStaff \Clef{octave=8} \TimeSig{4}{4}
  \Dotted{*/1,,*/3,} \Note{|-,=|,|-,=|}{1,2,3,4}
  \Note{4}{2}
  \Artic{^}{0} \Note{8}{0,1} \Tie[eshift=2]{1}
  \Barline
  \Note{1}{1}
  \Barline[t=|.]

  \LowerStaff \Clef[t=4] \TimeSig{4}{4}
  \Note{2}{13/10/8} \Space[1.5]
  \Dotted{13} \Dotted{9} \Dotted{7}
  \Note{4}{13/9/7} \Space[0.5]
  \Note{8}{12/10/<9/7}
  \Tie[eshift=2]{12}
  \Tie[bshift=0.25,eshift=1.75]{10}
  \Tie[swap,bshift=0.25,eshift=1.75]{9}
```

```

\Tie[swap,eshift=2]{7}
\Barline
\Note{1}{12/10/<9/7}
\end{mousik}

```

You may find longer examples in the GitHub repository for this package.

It has some options:

- *indent* indents the first line. Default is 0, in cm.
- *length* changes the maximum width of the score. It uses the package's internal length units. One unit corresponds to 0.12 cm (unscaled).
- *hsep* changes the base horizontal separation. Default is 8. It does not change *indent*, or the spacing of clefs, key signatures and time signatures.
- *accsep* changes the space for accidentals represented by `!`, in units of *hsep*. Default is 0.5.
- *barep* changes the horizontal space around barlines, in units of *hsep*. Default is 2, which centers a barline between two gaps.
- *scale* uniformly rescales the score, including symbols, text, line widths and spacing. Default is 1.
- *vsep* changes the vertical separation in case of an overflow. Default is 2, in cm.
- *single* draws a one-line staff, useful for percussion examples. Notes keep the usual absolute height system; height 6 lies on the line.
- *piano* draws an upper and a lower five-line staff. Use `\UpperStaff` and `\LowerStaff` to choose which one receives the following commands. Each staff keeps its own cursor, clef, and key.
- *piano-sep* changes the distance between the two piano staves. Default is 1.6, in cm.
- *piano-bars* chooses whether barlines in piano mode are joined across both staves or separate. Default is `joined`. The command advances only the active staff.
- *color* changes the background color. Default is white.
- *tikz* allows extra options for the outside `tikz` environment.

The package exports its commands globally with a `\mousik` prefix, for example `\mousikNote` and `\mousikText`. Inside the `mousik` environment, the shorter forms `\Note`, `\Text`, etc. are local aliases.

3 Available commands

3.1 `\Barline`

The `\Barline` command prints a bar on the score. It has some options:

- *t* (type) is the type of bar. Default is `|`, the single bar.

- *xshift* moves the element horizontally (x axis). Default is 0, in cm.
- *yshift* moves the element vertically (y axis). Default is 0, in cm.



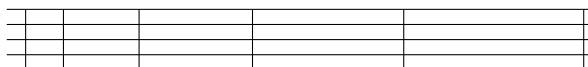
`\Barline t=||` `t=|.` `t=|:` `t=:|:` `t=:|` `t=:`

Automatic overflow is the safest option after a bar or a space. Otherwise, visual errors may appear (see subsection 4.1). To start a new line explicitly, use `\BreakLine`.

3.2 `\Space`

The `\Space` command may be used when no bar is needed, or to add extra space between symbols. Its optional argument changes the amount of space. Default is 1. One `\Space` unit is two *hsep* units, i.e. the cursor advance between events. Negative values move the cursor backwards, which can also be useful to superimpose voices or place symbols manually (see subsection 4.2).

For finer control, there are two related commands. `\NoteSpace` is measured in note units (one *hsep*). `\AccSpace` is measured in accidental-gap units (one !).



`[-0.5]` `[0]` `[0.5]` `[1]` `[1.5]`

3.2.1 `\BreakLine`

The `\BreakLine` command starts a new score line. It resets the horizontal cursor to the beginning of the next line. Unlike an automatic overflow, it does not print a clef or key. When `\BreakLine` is used on the first line, it also determines the staff length of the following lines.

3.3 `\Clef`

The `\Clef` command may be used at any moment on the score. A clef is also printed automatically after an overflow (see subsection 4.1). It has some options:

- *octave* prints an octave eight on the clef. If it is 8, the eight is printed above the clef; if it is -8, it is printed below the clef. Default is 0.
- *t* (type) is the type of clef. Default is 2 (the Treble Clef).
- *xshift* moves the element horizontally (x axis). Default is 0, in cm.
- *yshift* moves the element vertically (y axis). Default is 0, in cm.

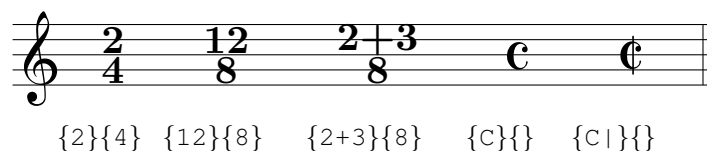


3.4 \TimeSig

The `\TimeSig` command prints a time signature. The first argument is the upper part and the second argument is the lower part. Any number or symbol can be written in them. `C` produces a *common time* symbol, and `C|` produces an *alla breve* symbol.

It has some options:

- `xshift` moves the element horizontally (x axis). Default is 0, in cm.
- `yshift` moves the element vertically (y axis). Default is 0, in cm.



3.5 Notes and rests

3.5.1 \Note

The `\Note` command prints notes and chords. It has two compulsory arguments: the duration and the height. The duration may be 1, 2, 4, 8, 16 or 32, and the height is absolute and in millimeters, starting from the middle C as 0.

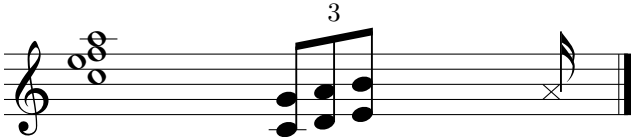


The second argument may also contain heights separated by `/` to form a chord, or positions separated by commas to form a group of notes. Each position in a

group may itself be a chord. To move the head of an item in a chord to the left, use `<`.

It has some options:

- *beam* chooses the beam width. Default is 1, in mm.
- *sloped* makes a beam of three or more notes follow a slope from the first note to the last instead of being horizontal. Two-note beams are always oblique.
- *head* chooses the type of head. Default is `.` (round), but `x` (cross) and `o` (hollow) are also possible.
- *stem* chooses the stem height. Default is 5, in mm. By default, stems point up below height 6 and down otherwise.
- *swap* reverses the automatic stem direction. Default is false.
- *tuplet* prints the number of a tuplet. The number is calculated by default, but it may also be assigned. *tuplet-bracket* adds a bracket, and *tuplet-xshift* and *tuplet-yshift* move the tuplet marking.
- *xshift* and *yshift* move the element horizontally and vertically. Default is 0, in cm.



```
\Note{1}{12/10/<9/7} \Note[head=x]{16}{5}
\Note[sloped,tuplet,swap,stem=6]
{8}{0/4,1/5,2/6}
```

Notes with different durations may also be joined in the same beam group. In this case the first argument is a list. `-` means an eighth note, `=` a sixteenth note and `≡` a thirty-second note. A `|` may be added before or after a symbol to mark where that beam level begins or ends. The forms `//` and `///` are equivalent to `=` and `≡`.

Inside a beam group, `!` inserts extra space for accidentals or dots. The extra amount is *accsep* note units, where one note unit is one *hsep*; by default *accsep*=0.5. An empty item leaves a whole *hsep* unit (usually for rests).



```
\Note{|≡, |=, |≡, ≡|, -, ≡|}{0,1,2,3,2,1}
```

3.5.2 `\Rest`

The `\Rest` command prints a rest. It accepts its duration as an argument: 1, 2, 4, 8, 16 or 32.

It has some options:

- *multi* transforms the rest into a multimeasure rest, where the rest lasts an arbitrary amount of bars.
- *xshift* moves the element horizontally (x axis). Default is 0, in cm.
- *yshift* moves the element vertically (y axis). Default is 0, in cm.



17

1 2 4 8 16 32

```
\Rest[multi]{17}
\Note{|-,,-,|}{3,,3}
\Space[-1.5]
\Rest[yshift=-0.2]{8}
\Space[0.5]
```

3.6 Keys and accidentals

3.6.1 `\Acc`

The `\Acc` command prints an accidental before a note. It accepts 2 arguments. The first argument is the type of accidental, from double flat to double sharp: (-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2). The flat $\flat$ symbol can also be called with a `b` and the sharp $\sharp$ symbol can be called with `\#`. The second argument is the height of the symbol, which corresponds to those of the notes.

It also has some options:

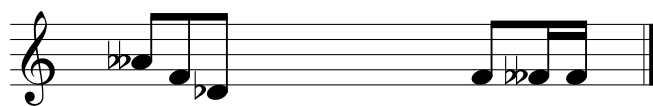
- *xshift* moves the element horizontally (x axis). Default is 0, in cm.
- *yshift* moves the element vertically (y axis). Default is 0, in cm.



-2 -1.5 -1 -0.5 0 0.5 1 1.5 2

b \#

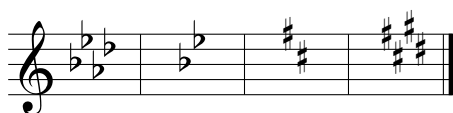
When there are several notes, the `\Accs` command may be useful. It accepts as an argument a list of pairs: type/height. Where there is no accidental, the spot can be left empty. It also accepts the *xshift* and *yshift* options.



```
\Accs{-2/5, , -1/1}      \Accs{, !, -2/3, }
\Note{8}{5, 3, 1}   \Note{|-, !, =, =|}{3, !, 3, 3}
```

3.6.2 \Key

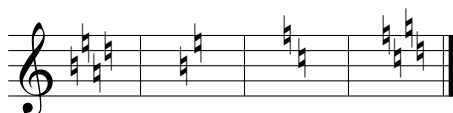
The `\Key` command prints the key at any moment on the score, although it is automatically printed with line overflows (see subsection 4.1). It accepts an argument: a positive whole number from 1 to 7 for sharps $\sharp$, and a negative whole number from -1 to -7 for flats $\flat$.



```
{-4}      {-2}      {2}      {4}
```

3.6.3 \NoKey

The `\NoKey` command cancels a key signature by printing naturals $\natural$ in the positions indicated by its argument. Positive values cancel sharps and negative values cancel flats. After `\NoKey`, the key signature is zero.



```
{-4}      {-2}      {2}      {4}
```

3.6.4 \MixedKey

The `\MixedKey` command prints a change of key signature. It accepts two arguments: the old key signature and the new one. Cancellation naturals are printed when needed.



```
{5}{2}      {2}{-3}
```

Be careful: the `\Key`, `\NoKey` and `\MixedKey` commands are affected by the current clef. Other commands such as `\Note` or `\Acc` are not affected and need absolute height values.



```
\begin{mousik}
  \Clef[t=4]
  \Key{1}
  \TimeSig{C|}{}

  \Acc{0}{8}
  \Note{1}{8}
  \Barline[t=|.]
\end{mousik}
```

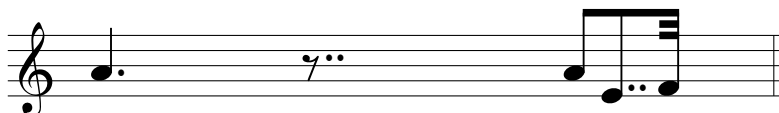
3.7 Other symbols

3.7.1 \Dotted

The `\Dotted` command adds one or two dots to the following note or rest. With one note, its compulsory argument is the height of the dot, which corresponds to those of the notes. With several notes, the argument may be a list of pairs: symbols/height, where symbols can be `*` or `**`. Where there are no dots, the spot can be left empty.

It has some options:

- `t` (type) accepts `**` or `..` to produce 2 dots when a single height is given. The default is `*`.
- `xshift` moves the element horizontally (x axis). Default is 0, in cm.
- `yshift` moves the element vertically (y axis). Default is 0, in cm.



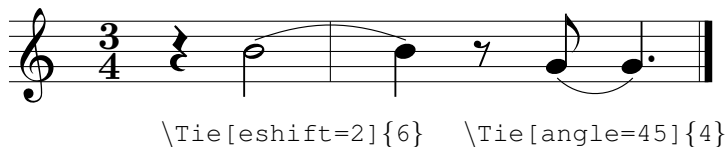
```
\Dotted{5} \Dotted[t=**]{7} \Dotted{, **/3, !, }
\Note{4}{5} \Rest{8} \Note{!-, -, !, \equiv}{5, 2, !, 3}
```

3.7.2 \Tie

The `\Tie` command draws a tie between two notes. It has a compulsory argument, the height of both notes, and some options:

- `angle` changes the angle in which it is drawn. Default is 20, in degrees.
- `bshift` changes the starting point. Default is 0, in cm.
- `eshift` changes the ending point. Default is 1, in cm.
- `swap` prints the element upside down. Default is false, which means like a *u* if the height is less than 6, and like an *n* otherwise.
- `width` changes the line width. Default is 0.15, in mm.
- `xshift` moves the element horizontally (x axis). Default is 0, in cm.

- *yshift* moves the element vertically (y axis). Default is 0, in cm.



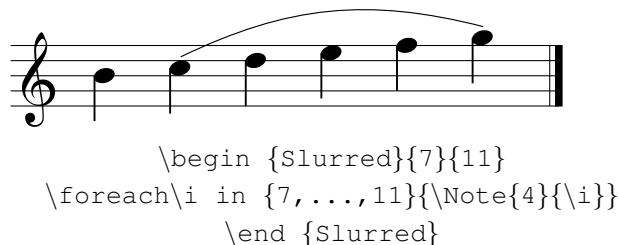
3.7.3 `\Slur`

The `\Slur` command draws a slur between two consecutive notes. It accepts two compulsory arguments: the absolute heights of its beginning and end. It accepts the same options as `\Tie`: *angle*, *bshift*, *eshift*, *swap*, *width*, *xshift* and *yshift*.



3.7.4 The `Slurred` environment

The `Slurred` environment draws a slur between the notes placed inside. It has two mandatory arguments: the heights of both beginning and end notes. It accepts the same options as `\Slur`: *angle*, *bshift*, *eshift*, *swap*, *width*, *xshift* and *yshift*.



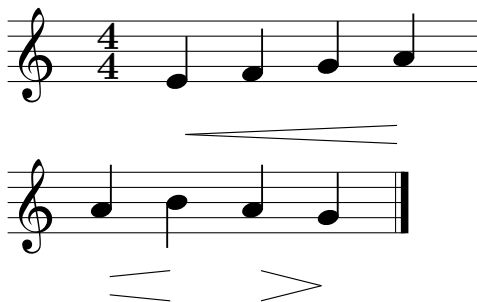
3.7.5 The `Crescendo` and `Decrescendo` environments

The `Crescendo` and `Decrescendo` environments draw hairpins between the notes placed inside. Their options are:

- *bopen* and *eopen* set the full opening of the hairpin at its beginning and end, in cm. A crescendo defaults to *bopen*=0, *eopen*=0.4. A decrescendo defaults to *bopen*=0.4, *eopen*=0.
- *bshift* and *eshift* move the beginning and ending points horizontally from the first and last heads. Defaults are 0.1 and -0.1, in cm.
- *width* changes the line width. Default is 0.15, in mm.

- *xshift* moves the hairpin horizontally. Default is 0, in cm.
- *yshift* moves the hairpin vertically. Default is 0, in cm.

A hairpin that continues after a line break should be written as two independent pieces. Give the first piece a chosen *open* and use the same value as *bopen* on the next line.



3.7.6 `\Artic` and `\Symbol`

The `\Artic` command prints an articulation symbol on a successive note. It accepts two arguments. The first one is the type of articulation. The second one is the height of the symbol, which corresponds to those of the notes—but it need not be the height of the corresponding note.

Staccatissimo		Marcato	Fermata	Mordent	Turn
{ ' }		{ ^ }	{ (.) }	{ ~ }	{ ? }

{ . }	{ > }	{ - }	{ + }	{ ~ }
Staccato	Accent	Tenuto	Trill	Lower mordent

Harmonic	Up bow	Down bow
{ o }	{ v }	{ n }

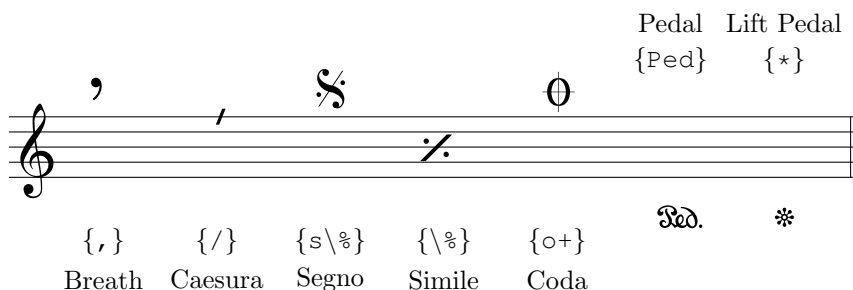
It has some options:

- *xshift* moves the element horizontally (x axis). Default is 0, in cm.
- *yshift* moves the element vertically (y axis). Default is 0, in cm.
- *swap* places the articulation on the opposite side of the note.

When there are several notes, the `\Artics` command may be useful. It accepts as an argument a list of pairs: symbols/height. Where there are no symbols, the spot can be left empty. It accepts the same *xshift*, *yshift* and *swap* options.



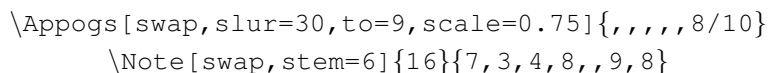
The `\Symbol` command is similar to the `\Artic` command, but it only accepts the type of symbol and not the height. These are symbols that have a fixed position on the score. It also accepts the *xshift* and *yshift* options.



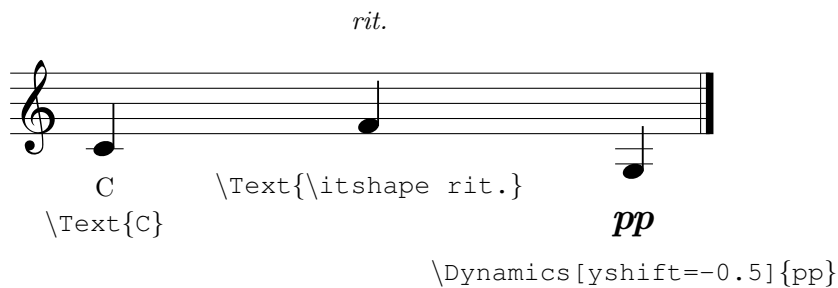
3.7.7 `\Appog`

The `\Appog` command prints an appoggiatura before the next note. Its two arguments are its duration (4, 8, 16 or 32) and its height. `\Appogs` admits a list of duration/height pairs. Both commands accept these options:

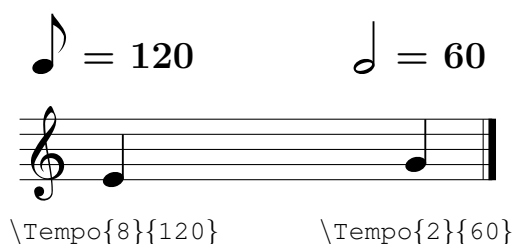
- *scale* changes the size of the appoggiatura. Default is 1.
- *swap* reverses the stem direction.
- *before* chooses how far before the main-note it is drawn, in cm. Default is 0.5.
- *xshift* and *yshift* work as expected.
- *slur* draws the optional appoggiatura slur and gives its angle in degrees.
- *to* gives the height of the main note to which the slur should attach.
- The slur is placed on the free side of the heads automatically. *slur-swap* reverses that choice.
- *slur-width* changes the slur line width; default is 0.12 mm. *slur-xshift* and *slur-yshift* move its target end, in cm.



The `\Text` command allows the printing of any text string. It has a compulsory argument: the text to be shown. The `\Dynamics` command works in the same way, but uses the usual font of dynamics. Both commands accept the *xshift* and *yshift* options.



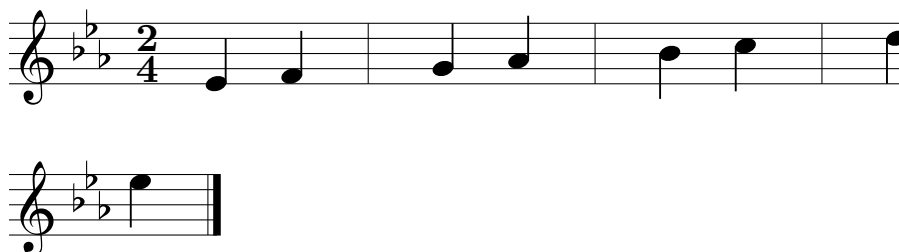
The `\Tempo` command prints a metronome tempo indicator. It has two arguments: the duration of the note and the numerical tempo which will be printed. It has the same options as the `\Text` command. Its default position is above the staff.



4.1 Overflow of lines

13

signature. For a more predictable layout you may use `\BreakLine`, add indents, change the length of spaces, or add or remove spaces for a more controlled positioning.



4.2 How do spaces work?

Spaces advance an invisible cursor while the score is rendered.

A negative argument moves the cursor backwards. For example, `\Space[-1]` moves it back by one normal separation. A `\Space` with 0 as the argument does nothing. Negative spaces can be used to superimpose voices for polyphony or to place symbols manually.

The `\Space` command is very relevant in this package. Because most layout logic is not implemented internally, the user may adjust each element as needed.

4.3 Using your own commands

If a particular rhythm is used repeatedly in a music piece, but the commands become increasingly long and tedious, the user may want to summarize it by creating their own vocabulary.



```
\newcommand{\ttc}[2][\Note[#1]{|-,|=,=|}{#2}}
\newcommand{\tct}[2][\Note[#1]{|=,=|,-|}{#2}}

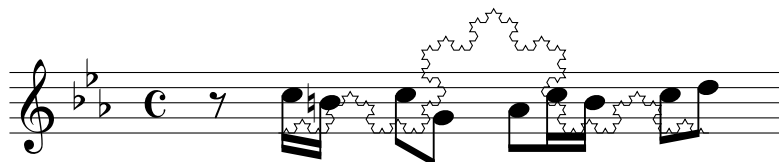
\begin{mousik}
  \Clef
  \ttc{7,8,9}
  \tct{10,9,8}
  \Barline[t=|.]
\end{mousik}
```

4.4 Working with the tikz package

The `mousik` package is based on the `tikz` package. In fact, the `mousik` environment is a disguised version of a `tikzpicture` environment, but with some initialized variables and a printed staff.

This implies two separate things:

1. Other commands may also be used inside the `mousik` environment, not just the commands explained in this text. For example, one may draw nodes, lines, or even a fractal behind my score:



2. The commands explained here may be used in a `tikzpicture` environment, and will appear with no staff. However, the command `\mousikInit` should be used at the beginning to avoid initialization errors.



5 Notes from the author

This package was developed almost entirely in the summer of 2020. It succeeds a smaller music notation package that I had developed in 2019 for twelve-tone music notation (`ddphonism`).

Back then, the hand-made computational geometry became very cumbersome, and some important features had display errors that were difficult to fix by hand (for instance, the placement of slurs, de/crescendos, and mixed keys). In 2026, generative AI tools were advanced enough to work through the remaining list of to-dos under my supervision.

The code relies on the `tikz` package for rendering, and the `musixtex` and `harmony` packages for musical symbols. The package is still in early development, so behavior may change in the future. I welcome any suggestions for improvement.