Guidedog Discussions GDS 0015

Engraved LilyPond music

STATE
committed
TYPE
Standards Track
AUTHORS
Vikrant Rathore
CREATED
2026-11-01
UPDATED
2026-11-01
DISCUSSION
No external discussion URL assigned
LABELS
architecture, parser, renderer, build, security

Record status

The catalog defines the review process and the meaning of each state. This numbered record forms part of the catalog.

Contents

1. Abstract 1
2. Motivation and scope 1
3. Specification 2
4. Rationale and alternatives 17
5. Security and operational considerations 18
6. Backwards compatibility 19
7. Acceptance criteria and implementation status 19
8. Open questions 19
9. Review history 20
10. References 21

1. Abstract

We add a `lilypond` directive to RST and MyST projects. Its body is music in LilyPond's input language. Guidedog engraves it itself, in Rust, without the LilyPond program, and draws it as SVG (or PNG) for HTML and PDF. Three new library crates do the work: `gd-score` holds a music model, `gd-lilypond` reads a documented subset of LilyPond into it, and `gd-engrave` lays it out and draws it. Horizontal spacing uses a spring and rod model solved exactly; lines break by a Knuth-Plass dynamic program; beams, slurs and ties are chosen by scoring candidate shapes; vertical collisions are resolved with skylines. Glyphs come from Bravura, the SMuFL reference font, embedded unmodified. Each builder engraves at its own measure, so the HTML page and the PDF book each get justified systems. Errors are Elm-style reports that point into the RST or Markdown file.

2. Motivation and scope

Documentation about music theory, instruments, audio software and composition needs music examples. Today an author engraves them elsewhere, exports images, and commits them. The images drift from their source, ignore the page's measure, and do not follow the theme's dark mode. GDS 0014 showed the value

of drawings that live in the document and build with it; Graphviz showed the same for graphs. We want the same for music, with the quality a reader expects from engraved scores.
LilyPond's input language is the natural source. It is plain text, it is widely known, it reads well in a diff, and many musicians already write it. LilyPond the program, however, is a GPL application with an embedded Guile Scheme interpreter. Its input is a programming language: since version 2.24 it no longer attempts a safe mode, so compiling an untrusted file can run arbitrary code (CVE-2020-17354). It cannot be linked into Guidedog's single executable, and running it per example costs about a second and an installed toolchain.
We therefore separate the notation from the engraver. We read the part of LilyPond's language that describes music (notes, rhythms, staves, voices, lyrics and common markings) and refuse the part that programs the engraver (Scheme, overrides and layout code). We independently implement algorithms from the published literature and LilyPond's documented behavior. Source references explain mathematical rules and interoperability checks; we do not translate or copy LilyPond implementation code into these crates.
In scope:

- The `lilypond` directive in RST and its MyST fence, with inline source or a file argument.
- The input profile of the section "The LilyPond profile", in three phases.
- Horizontal spacing, line breaking, stems, beams, chords, accidentals, dots, ties, slurs, tuplets, articulations, dynamics, lyrics, multiple staves and four voices per staff.
- SVG and PNG drawings for the HTML and PDF builders, adaptive to the theme in HTML.
- Diagnostics, limits and the trust policy.

Out of scope for this record:

- Running the LilyPond program, and Scheme, `\override`, `\tweak` and `\paper` code.
- MIDI or audio output, playback and score following.
- Percussion staves, figured bass, cross-staff notation, ancient notation, microtones, and more than four voices on a staff. Guitar tablature, first listed here, is added by GDS 0016.
- Inline music in a sentence (a role); see "Open questions".
- Writing LilyPond from other formats, and reading MusicXML, MEI or ABC. The model admits such readers later.

3. Specification

3.1. Overview

The pipeline has four representations. Each stage is a pure function of its input and options.



Fig. 18 From LilyPond text to drawings. Each box is a representation; each arrow is one stage owned by one crate.

The syntax tree keeps every token's file and byte span, so each later error can point at its source. The score is the music, independent of LilyPond: exact rational time, spelled pitches, staves, voices, spanners and lyrics. The engraving is geometry: systems, staves and a display list of glyphs, lines, polygons, curves and shaped text, in points. The writers turn the display list into SVG text; the build turns SVG into PNG when asked.

3.2. Directive

We write music as follows.

```
.. lilypond::
   :caption: The opening phrase
   :alt: A four-bar melody in G major
   :name: opening

   \relative c'' {
     \key g \major \time 3/4
     d4 g,8( a b c) | d4 g, g | e' c8( d e fis) | g4 g, g |
   }
```

The MyST fence begins with ```{lilypond} and carries the same colon options and body. An optional file argument supplies the whole source instead of the body; exactly one source is required. File arguments are retained during markup parsing and opened by the trusted link pass through the project's confined read policy. This keeps a text conversion faithful without engraving and lets an untrusted build refuse a music file before it is read. Ordinary code-block:: lilypond blocks and ```lilypond fences remain source examples.

Table 27 Directive options

Option	Meaning
alt	Alternative text. Without it, a description generated from the score is used (see "SVG").
caption, name, class, align	As for typst-diagram: figure caption and numbering, label, classes, and left, center or right.
format	svg (default) or png.
staff-size	The height of a five-line staff, in points, from 10 to 40. The default is the project's lilypond_staff_size, initially 18.
width	The measure, as a length (12cm, 400pt) or a percentage of the builder's measure. The default is 100%.
justify	auto (default), all or none; see "Line breaking".

The project gains three numeric settings, measured in typographic points. lilypond_staff_size is the default staff size. lilypond_html_width and lilypond_pdf_width override the builders' measures for themes and templates that change the column or the margins (see "HTML" and "PDF"). House style is otherwise fixed: every project engraves with the same rules and the same music font; only text faces follow the project's font policy.

3.3. The LilyPond profile

We name the accepted language the *Guidedog LilyPond profile*. Every accepted input means what it means to LilyPond 2.24 to 2.26; where Guidedog draws something differently, the difference is one of house style, never of meaning. Every input outside the implemented profile is refused with a report that names the construct and says what to do. The command table distinguishes implemented, planned, ignored and refused constructs. Planned commands are diagnosed as unsupported until their phase is complete; they must never be silently skipped. The implementation status lists the supported profile explicitly, and commitment requires all three phases. A few LilyPond leniencies become refusals; each is listed below.

A body is a *file* when its top level contains \version, \language, \include, \header, \layout, \score or an assignment. Otherwise it is a *fragment*: a sequence of music read as if wrapped in braces. LilyPond treats top-level music as an implicit score, and so do we: a fragment without \new Staff is one staff, and a fragment may contain << >>, \new Voice and \new Staff exactly as a score may. A file holds at most one \score block or one top-level music expression.

Table 28 Accepted constructs

Phase	Construct	Notes
1	Note names in nederlands (default), english and deutsch, selected by \language or by including english.ly and the other language files; is/es and doubled forms; octave marks ' and ,; octave checks =; forced ! and cautionary ? accidentals	Quarter tones are refused.
1	Durations 1 2 4 8 16 32 64 128 \breve, dots, and *n/m multipliers; a missing duration repeats the previous one, starting with a quarter	Exact rationals.
1	Rests r, spacers s, measure rests R, chords < >, chord repetition q	R must last a whole number of measures; LilyPond only warns.
1	\relative (with or without a start pitch), \fixed, \transpose	See "Interpretation".

Phase	Construct	Notes
1	`\clef` (`treble`, `bass`, `alto`, `tenor`, `soprano`, `treble_8`, `bass_8`), `\key` with `\major`, `\minor` and the church modes, `\time` with simple and compound meters and beat lists (`\time 2,2,3 7/8`), drawn as LilyPond draws them by default, with C for 4/4 and a C with a stroke for 2/2; `\numericTimeSignature` and `\defaultTimeSignature`; `\partial` at the start, bar checks `|`, `\bar` with `|`, `||`, `|.`, `.|`, `.|:`, `:|.`, `:...:`	`\time` off a bar line, `\partial` after the start, and different meters on different staves are refused.
1	Ties `~`, slurs `( )`, manual beams `[ ]`, `\autoBeamOff/On`, `\tuplet n/m`, `\times m/n`, `\stemUp/Down/Neutral`, `\slurUp/Down/Neutral`, `\phrasingSlurUp/Down/Neutral`, `\tieUp/Down/Neutral`, `\dynamicUp/Down/Neutral`	
1	Articulations `-. -- -> -^ -_ -! -+`, the named forms (`\staccato`, `\staccatissimo`, `\portato`, `\accent`, `\tenuto`, `\marcato`, `\stopped`, `\open`, `\flageolet`, `\thumb`, `\fermata`, `\trill`, `\turn`, `\mordent`, `\prall`), fingerings `-1` to `-5`, direction prefixes `^` and `_`	
1	Dynamics `\pppp` to `\ffff`, `\mp`, `\mf`, `\fp`, `\sf`, `\sfz`, `\rfz`; hairpins `\<`, `\>`, `\!`	
1	Text scripts `^"text"` and `_"text"`; structured LilyPond 2.24 `\markup` (lists, columns, styles, sizes, enclosures, spacing, music symbols and markup variables)	Scheme, layout and embedded-score markup commands are refused.
1	Lyrics of one verse: `\addlyrics`, `\lyricsto`, `\lyricmode`, hyphens `--`, extenders `__`, skips `_`, and quoted syllables	
1	Assignments of music, lyrics, strings or `\markup` (`melody = \relative c'' { ... }`) and their use (`\melody`)	No Scheme values.
1	`\include "file.ily"`	Trusted builds only; confined and tracked as a dependency.
1	`\midi { }`, `\header { ... }`, `\version "..."`, and an empty `\layout { }`	Accepted and ignored. `\header` fields do not print: in a document, the caption names the music. A non-empty `\layout` is refused with a hint naming the directive options.
1	`\break`, `\noBreak`	Forced and forbidden line breaks.
2	`<< >>` with `\\`, `\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`, `\oneVoice`, `\new Voice`, `\new Staff`, `\new PianoStaff`, `\new GrandStaff`, `\new StaffGroup`, `\new ChoirStaff`, `\with { instrumentName = "..." shortInstrumentName = "..." }`, `\set Staff.instrumentName` (string or `\markup`)	At most four voices sounding at once on a staff; a fifth is refused. `\shiftOn`, `\shiftOnn`, `\shiftOnnn` and `\shiftOff` set explicit column shift levels.
2	Several verses, `\new Lyrics` with its own durations, and phrasing slurs `\( \)`	
3	`\grace`, `\acciaccatura`, `\appoggiatura`, `\afterGrace`	
3	`\repeat volta n` with `\alternative` and `\volta` pass lists, `\fine` and `\section`, drawn with repeat bar lines and volta brackets; `\repeat unfold n`, expanded within limits; `\compressEmptyMeasures` and `\compressFullBarRests`	
3	`\tempo` with text, a metronome mark printed with a SMuFL note symbol, or both (`Allegro (quarter note = 120)`), `\mark \default`, `\mark "text"` and `\mark \markup ...`; `\breathe`	

Phase	Construct	Notes
3	`\chords` and `\new ChordNames` with `\chordmode` chord names in common jazz spelling, printed as LilyPond's default chord names: extensions raised, a triangle for a major seventh, a raised o or slashed o for diminished and half-diminished chords, and accidentals as music glyphs	

Scheme is accepted only as a literal value where the profile needs one: `##t`, `##f`, a number `#3`, a string `#"text"` and a symbol `#'up`. Everything else that begins with `#` or `$` is refused, as are `\override`, `\revert`, `\tweak`, `\once`, every other `\set`, `\paper`, `\book`, `\bookpart`, `\context` definitions and music functions defined in the file. Each refusal names the construct, places a caret under it, and says that Guidedog engraves a fixed house style.
`gd-lilypond` keeps one table of LilyPond commands, each marked accepted (with its phase), ignored or refused. We write it by hand from the command index of LilyPond 2.24's Notation Reference and revise it when the profile changes. A test checks that every accepted command has a reader test and every refused one a refusal test. The table also feeds suggestions, so a known but refused command is never reported as unknown.

3.4. Crates and their interfaces

Three library crates do the work. Dependencies point one way; the two outer crates do not know each other, so another reader or another writer can be added beside them.

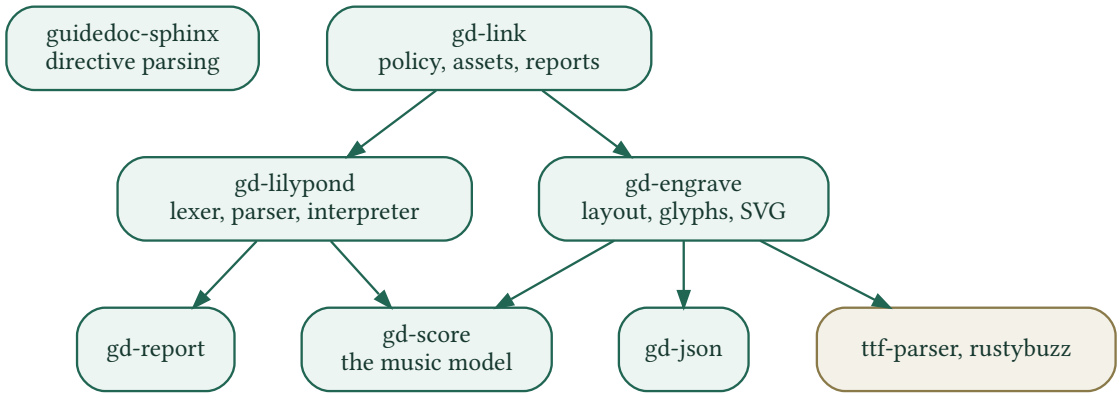

Fig. 19 Crate dependencies. Arrows point from a crate to what it uses.

All three crates set `publish = false` during the beta and follow the workspace rules: `version.workspace = true`, shared lints, no `unsafe`, no `unwrap` or `expect` on input-dependent values, functions of at most 70 logical lines, files of at most 1,408 lines, and a `///` comment on every public item saying what it borrows, returns and how it fails. They never print, hold no global state, and return values and structured errors.
The interface follows the project's raylib-inspired style. Options are plain structs with public fields and `Default`, filled with struct-update syntax. Verbs are few: `load_*` makes a value from text or bytes, `engrave_*` lays out, `write_*` serializes into a caller's buffer, `check_*` validates. Each crate has a one-call path for the common case beside the staged one. Identifiers are newtypes; choices are enums matched exhaustively; tables are built by `load_*` calls and immutable afterwards. Callbacks follow `gd-graphdog`: `Option<&'a mut dyn FnMut(...)>` fields named for what they decide.
`gd-score` has no dependencies. It also owns the shared source identifiers, source spans and diagnostic data, so the reader and engraver use the same vocabulary without depending on each other. The `gd-lilypond::error_report` adapter places these diagnostics through `gd-report`; the model itself does not depend on the report renderer. It is flat data: vectors of events and spanners indexed by newtype identifiers, not a graph of shared pointers.

```
/// An exact rational number, always reduced with `den > 0`. Arithmetic is checked;
/// `Ord` compares by cross-multiplication in i128, never field by field.
```

```rust
#[derive(Clone, Copy, Debug, PartialEq, Eq, Hash)]
pub struct Ratio { num: i64, den: i64 }
impl Ord for Ratio { /* num_a * den_b against num_b * den_a */ }

/// A point in musical time, in whole notes. `grace` orders grace notes before the
/// main moment they lead to (negative), as LilyPond's moments do.
#[derive(Clone, Copy, Debug, PartialEq, Eq, Hash, PartialOrd, Ord)]
pub struct Moment { pub main: Ratio, pub grace: Ratio }

pub struct StaffId(pub u32);
pub struct VoiceId(pub u32);
pub struct EventId(pub u32);
/// One note of a chord event: ties and lyrics attach to notes, not only to events.
pub struct NoteRef { pub event: EventId, pub note: u8 }

/// A spelled pitch. `octave` counts octave marks: `c` is 0 and `c'` (middle C) is 1.
/// (LilyPond's internal octave of `c'` is 0; we count marks so that tests read like
/// the source.)
pub struct Pitch { pub step: Step, pub alter: Alter, pub octave: i8 }
pub enum Step { C, D, E, F, G, A, B }
pub enum Alter { DoubleFlat, Flat, Natural, Sharp, DoubleSharp }

pub struct Duration { pub base: Base, pub dots: u8, pub scale: Ratio }
pub enum Base { Breve, Whole, Half, Quarter, Eighth, Sixteenth, ThirtySecond,
                SixtyFourth, HundredTwentyEighth }

pub struct Score {
    pub staves: Vec<Staff>,
    pub groups: Vec<StaffGroup>,
    pub voices: Vec<Voice>,
    pub events: Vec<Event>,
    pub spanners: Vec<Spanner>,
    pub lyrics: Vec<LyricLine>,
    pub measures: Vec<Measure>,
}

pub struct Event { pub voice: VoiceId, pub at: Moment, pub kind: EventKind, pub span: Span,
                   pub marks: Vec<Annotation>, pub voice_style: Option<VoiceStyle>,
                   pub column_shift: Option<u8> }
pub enum EventKind { Chord(Chord), Rest(Duration), Skip(Duration), MeasureRest(Duration),
                     Clef(Clef), Key(Key), Time(Meter), Bar(BarStyle), Break(BreakRule),
                     ChordName(String, Duration) }

/// A syllable is anchored to a note (`\addlyrics`, `\lyricsto`) or to a moment
/// (`\new Lyrics` with durations).
pub enum Anchor { Note(NoteRef), Moment(Moment) }

/// Checks the invariants every reader must establish: events sorted by moment within
/// a voice, spanners that start before they end, anchors that exist, one meter for
/// all staves. Borrows `score`; returns the first broken invariant.
pub fn check_score(score: &Score) -> Result<(), Error>;
```

`Span` is a `FileId` and a byte range; the score never holds LilyPond syntax. An overflow in `Ratio` arithmetic is an error (`score.time-overflow`), never a wrap or a panic.

`gd-lilypond` reads the profile. `\include` is lexical, so the parser, not the interpreter, calls the include callback.

```rust
/// What a parse asks for: the file it is in and the name it wrote.
pub struct IncludeRequest<'r> { pub from: FileId, pub name: &'r str }
/// What the caller read: a resolved name, for reports and dependencies, and its text.
pub struct IncludedFile { pub name: String, pub text: String }
/// Why the caller refused: one sentence and a hint.
pub struct IncludeRefusal { pub message: String, pub hint: String }
pub type Include<'a> =
    dyn FnMut(IncludeRequest<'_>) -> Result<IncludedFile, IncludeRefusal> + 'a;

/// What a load may do. Without `include`, every `\include` is refused.
#[derive(Default)]
pub struct Options<'a> {
    pub limits: Limits,
    pub include: Option<&'a mut Include<'a>>,
    pub reserve: Option<&'a dyn Fn(usize) -> bool>,
```

```
}

/// The text of every file read, indexed by `FileId`; file 0 is the given source.
pub struct SourceFiles { /* names and texts */ }

/// A score, the files read, and the warnings met (failed bar checks, ignored blocks).
pub struct Loaded { pub score: Score, pub files: SourceFiles, pub warnings: Vec<Error> }
/// A failed load keeps the files read, so its error can still be placed.
pub struct Failed { pub error: Error, pub files: SourceFiles }

/// One call: parses and interprets `source`, named `name`. Borrows both.
pub fn load_score(source: &str, name: &str, options: &mut Options<'_>)
    -> Result<Loaded, Failed>;

/// The stages, for tests and tools.
pub fn parse_lilypond(source: &str, name: &str, options: &mut Options<'_>)
    -> Result<Syntax, Failed>;
pub fn interpret_syntax(syntax: &Syntax, limits: &Limits) -> Result<Loaded, Failed>;
```

`gd-engrave` lays out and draws. Borrowed text faces carry an explicit collection index; reading every face of a collection into every style would mix regular, bold and italic text. `Fonts` is loaded once per build and shared by every thread: it borrows the font bytes and is `Sync`. An `Engraver` is a workspace: it keeps its buffers (columns, springs, skylines, candidate scores) between scores, so a page with many examples allocates once. One engraver serves one thread.

```
/// Text faces in fallback order, per style. The music font is built in.
#[derive(Default)]
pub struct FontOptions<'f> {
    pub regular: Vec<FaceData<'f>>,
    pub bold: Vec<FaceData<'f>>,
    pub italic: Vec<FaceData<'f>>,
    pub bold_italic: Vec<FaceData<'f>>,
}
/// One selected face, borrowing the original bytes; index 0 is the common case.
pub struct FaceData<'f> { pub data: &'f [u8], pub index: u32,
                          pub source: Option<&'f dyn FaceSource> }
/// Parses the embedded Bravura font and metadata and the given faces into immutable
/// tables. Borrows the faces for `'f`; fails when a face or the metadata is malformed.
pub fn load_fonts<'f>(options: &FontOptions<'f>) -> Result<Fonts<'f>, Error>;

pub type Reserve<'a> = dyn FnMut(usize) -> bool + 'a;
#[derive(Default, Clone, Copy)]
pub struct Scratch<'a> {
    pub reserve: Option<&'a (dyn Fn(usize) -> bool + 'a)>,
    pub release: Option<&'a (dyn Fn(usize) + 'a)>,
}

pub struct Options<'a> {
    /// The measure: the width of a system, in points.
    pub line_width: f64,
    /// The height of a five-line staff, in points.
    pub staff_size: f64,
    pub justify: Justify,
    pub optical_adjustment: f64,
    pub limits: Limits,
    /// Shown each buffer's growth in bytes before it happens; `false` refuses it.
    pub reserve: Option<&'a mut Reserve<'a>>,
    /// Scoped temporary layout buffers, reserved before allocation and released on drop.
    pub scratch: Scratch<'a>,
}
#[derive(Default)]
pub enum Justify { #[default] Auto, All, None }

#[derive(Default)]
pub struct Engraver { /* reusable buffers */ }
impl Engraver {
    /// Borrows `score` and `fonts`; returns systems of marks in points.
    pub fn engrave_score(&mut self, score: &Score, fonts: &Fonts<'_>,
                         options: &mut Options<'_>) -> Result<Engraving, Error>;
}
```

```
pub struct Engraving { pub width: f64, pub height: f64, pub systems: Vec<System>,
                       pub glyphs: BTreeMap<Glyph, GlyphData>,
                       pub summary: Summary, pub warnings: Vec<Error> }
pub struct System { pub top: f64, pub width: f64, pub height: f64,
                    pub first_measure: u32, pub last_measure: u32,
                    pub marks: Vec<PositionedMark> }
pub struct PositionedMark { pub mark: Mark, pub staff: StaffId,
                            pub measure: u32, pub bounds: Rect }
pub enum Mark {
    Glyph { glyph: Glyph, x: f64, y: f64, size: f64, role: Role },
    Line { from: Point, to: Point, thickness: f64, role: Role },
    Polygon { points: [Point; 4], role: Role },
    Curve { outer: Bezier, inner: Bezier, role: Role },
    Text { run: TextRun, x: f64, y: f64, role: Role },
}
/// The facts a description is made from; gd-link words them in the page's language.
pub struct Summary { pub staves: u32, pub instruments: Vec<String>, pub key: Option<Key>,
                     pub meter: Option<Meter>, pub measures: (u32, u32), pub lyrics: bool }

pub struct SvgOptions<'a> {
    /// The systems to write, by index; all of them by default.
    pub systems: Option<std::ops::Range<usize>>,
    /// The `<title>` and `<desc>` text.
    pub title: &'a str,
    pub description: &'a str,
    /// Adds the screen stylesheet with the dark ink; for HTML only.
    pub adaptive: bool,
    pub max_bytes: usize,
}
/// Writes the chosen systems of `engraving` as one SVG document into `out`. Fails
/// on invalid geometry, missing glyphs, output limits or refused buffer growth.
pub fn write_svg(out: &mut String, engraving: &Engraving, options: &SvgOptions<'_>,
                 reserve: Option<&mut Reserve<'_>>) -> Result<(), Error>;

/// One call: engraves `score` with fresh buffers and returns all of it as SVG.
pub fn engrave_svg(score: &Score, fonts: &Fonts<'_>, options: &mut Options<'_>)
    -> Result<String, Error>;
```

Both crates report problems with one shape, close to `gd-graphdog`'s but with spans, since a score can come from several files.

```
pub struct Error {
    pub kind: ErrorKind,     // exhaustive enum with a stock code, title and hint
    pub severity: Severity,  // Error or Warning
    pub span: Span,          // FileId and byte range; empty when not in the source
    pub message: String,     // one sentence
    pub hint: String,        // how to fix it, with "Did you mean ...?" when we can tell
}
/// Places a shared diagnostic using the source files retained by the reader.
pub fn error_report(error: &Error, files: &SourceFiles) -> gd_report::Report;
```

Source spans use byte offsets at UTF-8 boundaries and include an explicit file identifier. Includes preserve the including file when resolving nested paths and are checked for cycles before expansion. The source limit counts repeated expansions, not only distinct files. Diagnostics use a retained per-line source map when indentation differs or tabs occur; a single indentation count is sufficient only for uniformly indented bodies.

A directive's body is not the RST file: the parser strips its indentation. `guidedoc-sphinx` therefore records, beside the body, a per-line source offset map in the custom attribute `lilypond-map` (also for MyST fences). `gd-link` maps a position in the body to the document through `lilypond-map`, so a report's line, column and caret are exact in the RST or Markdown file. A position in an included `.ly` file is reported in that file.

The adapters stay small, but they touch the same places `typst-diagram` touches:

- `guidedoc-sphinx`: `lilypond.rs` parses the directive and `directives.rs` registers it.
- `guidedoc-rst-writer`: `literal.rs` writes it back.
- `gd-read`: `collect` classifies a captioned drawing as a figure.

- `gd-link`: `page.rs` dispatches the code block to `lilypond.rs`, which owns policy, limits, fonts, assets, localized descriptions and reports, and `figure.rs` gains the narrow edition, system parts and a generated alternative text.
- `guidedoc-html`: an image with a narrow edition is written as `<picture>`.
- `guidedoc-typst`: an image of several system parts is written as a breakable block, and a music image wider than the column is scaled to fit.
- `gd-theme`'s Typst templates: each declares its horizontal margin for the PDF measure.
- `gd-typst`: one function that rasterizes an SVG at a given resolution, and one that returns the text faces its font policy selects.
- `gd-theme` and `gd-i18n`: the narrow-screen rules and the words of generated descriptions.

The implemented modules are:

Table 29 Implemented modules

Crate	Modules
`gd-score`	`ratio`, `pitch`, `duration`, `markup`, `score` (including `Moment`), `check`, `error`
`gd-lilypond`	`lexer`, `cursor`, `syntax`, `parser`, `names` (note-name languages), `commands` (the command table), `bindings`, `markup`, `notation`, `notes`, `contexts`, `lyrics`, `repeats`, `ornaments`, `interpret`, `finish`, `files`, `memory`, `limits`, `error`
`gd-engrave`	`prepare`, `columns`, `spacing` (springs, rods, solver), `rods`, `breaking`, `layout`, `geometry`, `accidentals`, `courtesy`, `beaming` (groups), `beams` (quanting), `beam_segments`, `heads`, `dots`, `rests`, `multirests`, `voices` (note-column shifts, rests), `curves`, `bows`, `hairpins`, `tuplets`, `decorations`, `marks`, `signatures`, `bars`, `groups`, `labels`, `lyrics`, `footprints`, `skyline`, `vertical`, `fonts` (SMuFL tables, text faces), `text`, `markup`, `drawing`, `svg`, `math`, `memory`, `scratch`, `limits`, `error`

3.5. Interpretation

Interpretation turns the syntax tree into the score. It resolves pitches and durations, places every event at an exact moment, assigns events to voices and staves, attaches lyrics, and checks bars.

Pitches. Let a pitch's diatonic index be $n = 7o + j$, where o is the octave and $j \in \{0, \ldots, 6\}$ the step from C. In `\relative` mode, a note written with step j and k octave marks (`'` counts $+1$, `,` counts -1) after a reference with index n_0 takes the unique index

$$n = n_0 + d + 7k, \qquad d \equiv j - (n_0 \bmod 7) \pmod 7, \qquad -3 \le d \le 3.$$

This is LilyPond's rule that the interval to the previous note is less than a fifth; the written accidental does not affect the choice. The next reference is the new note. Inside a chord, each note is relative to the one before it; after a chord, the reference is the chord's first note. Without a start pitch, the reference is `f` (index 3), which makes the first note read as an absolute pitch, as LilyPond documents. `\relative` does not reach into a nested `\relative`, `\transpose` or `\chordmode`, as in LilyPond. `\fixed p` adds the octave of `p` to every note.

Transposition. `\transpose a b` defines an interval of $\Delta n = n_b - n_a$ diatonic steps and $\Delta c = c(b) - c(a)$ semitones, where c is a pitch's chromatic number. A pitch p becomes the pitch p' with $n_{p'} = n_p + \Delta n$ and the alteration that makes $c(p') = c(p) + \Delta c$. Spelling is therefore preserved: C major transposed from `c` to `ees` has three flats, never three sharps. `\key` is transposed with its tonic, so a key such as G sharp major can result; the key signature then holds a double sharp, as LilyPond draws it. An alteration beyond a double sharp or double flat is an error with a hint to choose an enharmonic target; LilyPond respells it with a warning, but a respelling the author did not write is a silent change of meaning.

Time. Every duration is an exact `Ratio`: the base value times $2 - 2^{-\,\mathrm{dots}}$, times tuplet and `*n/m` scale factors. `\partial d` starts the first measure at $-d$. Measures are cut at each multiple of the meter's length, counted from the last `\time` change. A bar check at a moment that is not a measure boundary is a warning, as it is in LilyPond, and engraving continues; the report says how far into which measure it fell. A body with no music is an error (`lilypond.empty`); a voice without events is dropped, and a staff of spacers is drawn empty.

Voices. `<< A \\ B >>` creates voices one and two on the current staff. Voice one has stems up and voice two stems down unless an explicit direction is set. Events in different voices at the same moment share a column; they do not need to agree in duration.

Lyrics. `\addlyrics` and `\lyricsto` anchor syllables to the notes of a voice in order, skipping tied continuations, notes under a slur and, when automatic beaming is off, notes under a manual beam (melismata), as LilyPond does. Rests take no syllable. A syllable with `--` gets a hyphen to the next syllable; one with `__` gets an extender to the end of its melisma. Syllables left over when the notes run out are dropped with a warning. Accidentals and beam groups are notation decisions, so the engraver makes them (see below). The score records only forced and cautionary accidentals, manual beams and whether automatic beaming is on.

3.6. Horizontal spacing

The engraver first builds *columns*: one for each distinct moment at which any staff has an event, plus columns for clefs, key and time signatures and bar lines. Grace notes (phase 3) get columns at their grace moments, just before their main moment. Spacing then has two parts: what each gap would ideally be, and what it must at least be.

Ideal gaps (springs). Space grows with duration, but slowly: doubling a duration adds a fixed amount of space rather than doubling it. The engraving manuals give tables of proportions with this property, Gould's among them; Gourlay stated it as a logarithmic rule, and LilyPond's `Spacing_options::get_duration_space` implements his rule with a linear branch for very short notes. We adopt LilyPond's function and defaults. Let $u = 1.2$ staff spaces (`spacing-increment`, about a notehead's width) and $\kappa = 2$ (`shortest-duration-space`). Let s be the score's *common shortest duration*: the most frequent value among the measures' shortest durations, the later on a tie, but never longer than $3/16$ (LilyPond's `base-shortest-duration`). For a duration δ we define

$$\varphi(\delta) = \begin{cases} u(\kappa + \log_2(\delta/s)) & \text{if } \delta \geq s, \\ u(\kappa - 1 + \delta/s) & \text{if } \delta < s. \end{cases}$$

Both branches agree at $\delta = s$, and a note half as long as s gets $u(\kappa - \frac{1}{2})$, not half the space. Let column i be at time t_i, and let δ_i be the shortest duration of any note or rest that starts at column i. When the next column comes sooner, because another voice or staff moves, the gap gets its share of the space:

$$\ell_i = \varphi(\delta_i)\,\frac{t_{i+1} - t_i}{\delta_i}.$$

An optical correction follows: a gap from an up-stem to a down-stem widens, and one from a down-stem to an up-stem narrows, by a tunable amount (initially 0.4 staff space). Clefs, key and time signatures and bar lines have fixed ideal gaps from the house style. The common shortest duration is computed once for the whole score, so ideal gaps do not depend on where lines break.

Minimum gaps (rods). Each column has, for each staff and each vertical band of a quarter staff space, how far its content (noteheads, accidentals, dots, flags, articulations, lyrics) reaches left and right of the column's reference line. For each staff we take the columns in which that staff has content, in order, and require consecutive ones to clear each other by a padding of 0.2 staff space in every band. Bands are compared one by one, so a low accidental can tuck under a high notehead in the previous column. Lyrics add rods: syllables of a line must be at least a word space apart, and a hyphen needs room for itself. A rod between columns i and $j > i + 1$ is a *spanning rod*; it arises when a staff or a lyric skips columns that other staves use.

Before solving, each spanning rod $x_j - x_i \geq r_{ij}$ is converted into adjacent rods: if the adjacent rods $r_i, \ldots, r_{j-1}$ sum to less than r_{ij}, each is raised in proportion to its ideal gap until the sum reaches r_{ij}. Raising never lowers a rod, so a spanning rod satisfied once stays satisfied; we process spanning rods by increasing span, then from left to right, which makes the result deterministic. The conversion is conservative: every solution of the converted problem satisfies every original rod.

Solving a line. Give spring i the stiffness $1/\ell_i$, so that all springs stretch in proportion. For a stretch factor $\lambda \geq 0$, gap i is

$$d_i(\lambda) = \max(r_i, \lambda\ell_i), \qquad D(\lambda) = \sum_i d_i(\lambda).$$

D is continuous, convex, piecewise linear and non-decreasing, with breakpoints $\lambda_i = r_i/\ell_i$. For a line of free width W (the measure less the prefatory matter), we sort the breakpoints and find the smallest λ^* with $D(\lambda^*) = W$ in $O(m \log m)$ time for m gaps. If $W < \sum_i r_i$, the line is *overfull*.

The configuration this produces is not a heuristic: it is the unique minimizer of the spring energy

$$\mathcal{E}(d) = \sum_i \frac{(d_i - \ell_i)^2}{2\ell_i} \quad \text{subject to} \quad d_i \geq r_i, \quad \sum_i d_i = W.$$

With multipliers $\mu_i \geq 0$ for $d_i \geq r_i$ and F for the sum, the Karush-Kuhn-Tucker conditions are $(d_i - \ell_i)/\ell_i = F + \mu_i$ and $\mu_i(d_i - r_i) = 0$. Where the rod is slack, $d_i = (1+F)\ell_i$; where it binds, $d_i = r_i \geq (1+F)\ell_i$. Together, $d_i = \max(r_i, (1+F)\ell_i)$: the same family with $\lambda = 1 + F$. Since $\mathcal{E}$ is strictly convex and the constraints are linear, the minimizer is unique.

3.7. Line breaking

Lines may break at bar lines, except where `\noBreak` forbids it or a beam crosses the bar; `\break` forces a break. A line from break a to break b needs prefatory matter at its start (clef and key signature, the time signature where it changes, and on the first line the indent for instrument names) and courtesy signatures at its end when the next line changes clef, key or meter. Ties, slurs, hairpins and extenders that cross a break are split into two halves; a tied note continued on a new line shows its accidental again. A hyphen that would fall at a break is drawn at the end of the earlier line. These additions depend only on a and b, so the springs and rods inside a line are computed once and reused for every candidate.

Each candidate line has a *badness* from its stretch factor. It is cubic, as in TeX, and grows without bound as a line is compressed toward nothing:

$$\beta(\lambda) = \begin{cases} 100\,(\lambda - 1)^3 & \text{if } \lambda \geq 1, \\ 200\,(1/\lambda - 1)^3 & \text{if } 0 < \lambda < 1. \end{cases}$$

Near $\lambda = 1$, $1/\lambda - 1 \approx 1 - \lambda$, so a small compression costs twice as much as the same small stretch; as $\lambda \to 0$ the cost grows without bound, so a crushed line never beats a loose one. An overfull line is a candidate only when it holds a single measure or ends at a forced `\break`; it is then set at its rods with badness 10^4, and the build warns (`engrave.overfull`). This guarantees that a breaking always exists.

Following Knuth and Plass, and Hegazy and Gourlay's application of their method to music, the demerits of a line ending at break b with penalty p_b are

$$\text{demerits}\ (a, b) = \begin{cases} (q + \beta)^2 + p_b^2 & \text{if } p_b \geq 0, \\ (q + \beta)^2 - p_b^2 & \text{if } p_b < 0, \end{cases}$$

where $q = 10$ is a line penalty. An ordinary bar line has $p_b = 0$; a double bar, a repeat sign or a change of key or meter has $p_b = -20$, so a break there is preferred, as Gould advises. The cost $C(b)$ of the best breaking up to b satisfies

$$C(0) = 0, \qquad C(b) = \min_{a<b}(C(a) + \text{demerits}\ (a, b)),$$

with a forced break forbidding candidates that cross it. A line may start at any eligible break after the previous forced break; forcing an endpoint must not prohibit earlier lines in that interval. A candidate line stops growing once its rods alone exceed W (or at the line limit of 128 measures), so each break looks back over a bounded window of w measures, and breaking costs $O(nw\,m \log m)$ for n measures. Before breaking, we compare $\sum r/W$ with the system limit, so a score that cannot fit is refused before the dynamic program runs. Within the declared 128-measure candidate window, the result is the best admissible set of breaks, not a greedy one: a loose line early can be avoided by a tighter one later.

The last line is treated as TeX treats a paragraph's last line, and so is a score that fits on one line. With `justify: auto`, a last line whose natural width ($\lambda = 1$) fits is set at its natural width and adds no badness; a short example is therefore drawn at its natural width. `justify: all` stretches every line to the measure. `justify: none` breaks optimally but sets every line at $\min(1, \lambda^*)$.

3.8. Vertical layout

Staff positions. A note's staff position is $p = n - n_c$ half spaces from the middle line, where n_c is the diatonic index of the clef's middle-line pitch. Positions with $|\ p\ | \geq 6$ need ledger lines, which extend beyond the notehead by the font's `legerLineExtension`.

Stems. In a single voice, a chord's stem goes down when its extreme notes lie more above the middle line than below ($p_{\max} + p_{\min} > 0$), up when the reverse holds, and down when they balance, following Gould and LilyPond. In a beamed group, the note farthest from the middle line decides for the group; a tie is

broken by the majority of notes, then down. Voices one and three point up; voices two and four point down. A local voice policy belongs to each event, so changing it later does not rewrite earlier music; `\oneVoice` restores automatic single-voice directions. A chord's optional direction overrides that policy, including an explicit neutral direction from `\stemNeutral`; changing stems does not erase the inner-voice column policy. A stem reaches 3.5 staff spaces beyond the notehead nearest its tip; it is lengthened to reach the middle line for notes far outside the staff, and for three or more flags. Stems attach at the notehead's SMuFL anchors `stemUpSE` and `stemDownNW`.

Chords. Noteheads a second apart sit on opposite sides of the stem. We walk the chord from the stem's base toward its tip; a note a second from the previous note, when that note is on the normal side, goes to the other side. For an up-stem the normal side is the left, and for a down-stem the right. Dots sit in spaces: a note on a line has its dot in the space above, or below for voices two and four, and clashing dots move to the nearest free space.

Voices sharing a staff. Voices sharing a staff share columns. Two noteheads at the same pitch share one head when their head glyphs and dots agree; otherwise, as for a half note against a quarter or a dotted against an undotted note, voice two's note moves right by a notehead width. In the two-voice case, an uncrossed second shifts the lower, downward-stemmed voice right; when the voices cross, the upward-stemmed voice shifts right. A rest in voice one is raised to staff position 4 and one in voice two lowered to position -4, then moved further, by whole staff spaces, until it clears the other voice's skyline. With three or four voices, we place each column against all previously placed columns, in voice-policy order. Each column has a shift level, LilyPond's `horizontal-shift`: one for voices three and four, zero for the others, or the explicit level of `\shiftOn` (1), `\shiftOnn` (2), `\shiftOnnn` (3) or `\shiftOff` (0). We place columns in order of level, then voice policy. A column starts half a notehead width aside for each lower level among earlier columns with the same stem direction, as LilyPond's note collisions shift it: right for upward stems, left for downward ones; columns with equal levels start together. From there we try at most sixteen notehead-width steps in the same direction. Candidate columns must clear other heads, stems and unattached flags; a compatible shared head has one upward and one downward stem. This also applies when only two of those voices sound at a moment. Accidentals and dots are then arranged across the whole staff column. Rests of equal duration at one moment are drawn once, using the earliest matching voice as owner, when no other voice sounds there. Where shifted voices share a column, its heads form one unit for spacing, so every head of the next column stands right of all of them. Measure rests, tuplet brackets, articulations, fingerings and text scripts follow their voice's side, and scripts follow its shifted heads; overlapping marks of different voices stack outward. The drawing records which marks form one sign, such as a dynamic's letters, a tuplet's digits or a hairpin's two lines, so stacking moves whole signs and keeps kerned letters together. Dynamics stay below unless `\dynamicUp` asks otherwise, because LilyPond's polyphonic directions do not include dynamics. In polyphony a slur rises at most 2.5 staff spaces to clear other voices and may cross them, as LilyPond's slurs avoid only their own voice; a tie rises at most 1.5. Other rests move by whole staff spaces away from notes and previously placed rests, with a maximum of thirty-two trials and a clear refusal if they cannot fit. Same-side tie endpoints start or end beyond the actual stem edge.

Accidentals. An accidental is shown when a note's alteration differs from the staff's current state for that step and octave. The state starts from the key signature at each measure and remembers accidentals until the bar line. A tied continuation shows none, except at the start of a line; `!` forces one and `?` adds parentheses. A key change mid-piece prints naturals that cancel the accidentals the new key drops, as LilyPond does by default. In a chord, accidentals are placed in columns to the left, in Gould's order: the highest, then the lowest, then the second highest, then the second lowest, and so on, with octaves kept in one column when they fit. Each accidental takes the column nearest the chord where its outline does not touch what is already there; outlines are bounding boxes refined by the SMuFL cut-outs (`cutOutNE`, `cutOutSW` and the others), so a flat can nest below a sharp.

Clefs. A clef change mid-line is drawn with the smaller change glyph (`gClefSmall` and its relatives from Bravura's optional-glyph metadata) before the note it applies to. A clef change at the start of a line also prints a courtesy clef at the end of the previous line.

Beam groups. With automatic beaming, notes shorter than a quarter are grouped by the meter's beat structure, and a rest ends a group. We write our own table from engraving convention and check it against LilyPond's default time-signature settings: by beat by default; eighths across the whole bar in 3/4 and 3/8;

eighths by the half bar in 4/4 and 2/2, with shorter notes by beat; and dotted beats in compound meters (6/8, 9/8, 12/8). A beat list such as `\time 2,2,3 7/8` sets the groups explicitly; without one, 5/8 groups as 3+2 and 8/8 as 3+3+2. A tuplet that fills a beat is one group. Manual `[ ]` beams override the automatic groups.

Beam positions. A beam's position is chosen from a finite set of candidates by score, as in LilyPond's beam formatting and in line with Gould's rules. Let the stems' natural ends give an ideal left height y_L^0, an ideal right height y_R^0 and an ideal slope θ_0, clamped to a maximum that grows with the beam's length, and zero for concave groups (an inner note nearer the beam than both outer notes). Candidates are all pairs (y_L, y_R) on a quarter-space grid within two staff spaces of the ideal, at most $17 \times 17 = 289$ pairs. Each candidate costs

$$B(y_L, y_R) = w_{\text{stem}} \sum_j \left(\frac{L_j - L_j^0}{L_j^0} \right)^2 + w_{\text{slope}} \, (\theta - \theta_0)^2 + w_{\text{quant}} (Q(y_L) + Q(y_R)) + w_{\text{short}} \, S + w_{\text{collide}} \, K,$$

where L_j are the resulting stem lengths and L_j^0 their ideals; Q penalizes end positions inside the staff that are not *sitting* on, *straddling* or *hanging* from a staff line, because such positions leave thin white wedges between beam and line; S counts stems shorter than the minimum for their beam count; and K counts collisions with noteheads, accidentals and inner beams. We take the cheapest candidate, breaking ties toward the smaller slope and then the lower position. Secondary beams lie `beamSpacing` apart; a fractional beam points into its own beat. The weights are named constants, calibrated against the reference corpus.

Tuplets. A tuplet always shows its number, centred on the beam side. Its bracket is omitted when the tuplet is exactly one beamed group, as LilyPond's default does, and drawn otherwise.

Ties. A tie joins two noteheads of the same pitch. It curves away from the stem; in a chord the upper half curves up, the lower half down, and the middle note of an odd chord away from the stem. Its ends start just beside the noteheads, and its apex is moved to the middle of a staff space when it would otherwise touch a staff line.

Ties and slurs share one bow shape, LilyPond's `slur_shape` in `lily/bezier-bow.cc`. For a span w, a height limit h_∞ and a ratio ρ, the inner control points lie at height

$$h(w) = h_\infty \, \frac{2}{\pi} \arctan\!\left(\frac{\pi \rho w}{2 h_\infty} \right),$$

nearly ρw for short bows and approaching h_∞ for long ones. They are inset from the ends by

$$e(w) = 2h_\infty - \frac{f \, g^2}{w + g}, \qquad g = \frac{2h_\infty}{f}, \qquad f = \frac{1}{3.1},$$

so that $e(0) = 0$, $e'(0) = f$, slightly under one third, the largest indent that keeps the bow from bulging past its ends, and $e \to 2h_\infty$ for long bows, which keeps their ends from looking hooked. The curve is the cubic Bezier from $P_0 = (0, 0)$ through $P_1 = (e, h)$ and $P_2 = (w - e, h)$ to $P_3 = (w, 0)$, rotated into place; its apex, at $t = \frac{1}{2}$, is $\frac{3}{4}h$. We begin with LilyPond's defaults: $h_\infty = 1$ staff space and $\rho = 0.333$ for ties, $h_\infty = 2$ and $\rho = 0.25$ for slurs.

A tie or slur is drawn filled, as engravers draw them: thin at the ends and thicker in the middle. The outer and inner curves share the construction, with ends τ_e apart and inner control points Δh apart. At the apex the thickness is $\frac{1}{4}\tau_e + \frac{3}{4}\Delta h$, so for a midpoint thickness τ_m we take

$$\Delta h = \frac{4\tau_m - \tau_e}{3},$$

with τ_e and τ_m from the font's `tieEndpointThickness`, `tieMidpointThickness`, `slurEndpointThickness` and `slurMidpointThickness`.

Slurs. A slur is chosen by score from candidates, as LilyPond does. Its ends are tried at the noteheads, or at the stem tips where stems point toward the slur, on a quarter-space grid of offsets around each attachment point (at most 49 pairs). Each candidate takes its height from $h(w)$, then rises just enough to clear the skyline of what it encloses, found by sampling the curve at 16 points. Its cost adds the distance of its ends from their attachment points, the difference between its slope and the notes' slope, the extra height it needed, the variance of its distance from the enclosed noteheads, and a penalty when it runs close to a tie. The cheapest candidate wins. A slur may cross staff lines; only tie apexes avoid them.

Measure rests. `R` of one measure is a whole rest centred in its measure (a breve rest in meters of four halves or more). Consecutive measure rests (`R1*4`) are drawn as one multi-measure rest: a thick horizontal bar with serifs (the SMuFL `restHBar` glyphs) and the count above, in a measure of fixed ideal width. It is broken across lines only at the measures it contains, which then become separate multi-measure rests.
Skylines. Each staff keeps an upper and a lower skyline: a piecewise-linear function of x giving how far its content reaches. Slanted beams and curves enter as sloped segments rather than boxes. Merging two skylines, and the least vertical distance at which two skylines clear each other, are both linear in their segment counts. Objects outside the staff are placed in a fixed priority order, nearest the staff first: articulations, fingerings, tuplet numbers and brackets, slurs, dynamics and hairpins, text scripts, then (phase 3) tempo marks, rehearsal marks and volta brackets. Each takes the position nearest the staff that clears the skyline by a padding of 0.25 staff space, and is then added to it. Dynamics and hairpins joined in one passage share a baseline, as Gould asks.
Staves and systems. Adjacent staves are placed as close as their facing skylines allow with a padding of one staff space, but never closer than nine staff spaces between middle lines. Lyric lines take the space between their staff and the next, sized by the text face's ascent and descent. A piano or grand staff gets a brace, a staff group a bracket, and connected bar lines follow the grouping. Instrument names are set at the left of the first system, and short names on the others; the first system is indented by the longest name. Each system after the first shows its first measure's number above the top staff, as LilyPond and most editions do. Systems are stacked with a padding of two staff spaces between facing skylines.

3.9. Glyphs and text

Music symbols come from Bravura, the reference font of the Standard Music Font Layout (SMuFL), published by Steinberg under the SIL Open Font License 1.1. We vendor the unmodified `Bravura.otf` and `bravura_metadata.json` of one pinned release under `third_party/bravura` with their license, and embed both in the executable (about 2 MB). Embedding unmodified files keeps us clear of the license's rules for modified versions and its reserved font name, and keeps the door open for other SMuFL fonts. `load_fonts` reads the metadata with `gd-json` into immutable tables: `engravingDefaults` (line, stem, beam, tie and slur thicknesses, beam spacing, ledger-line extension), each used glyph's bounding box, anchors and cut-outs, and its outline from the font. This takes a few milliseconds once per build; there is no lazily built global table. A test checks that every glyph the engraver names exists in the tables, and `cargo xtask` gains a check that the vendored files match their recorded digests. SMuFL units are staff spaces, so the tables need no scaling until the drawing is written.
Text (lyrics, instrument names, numbers in text, text scripts and markup) is shaped and turned into outlines in the engraver. Its faces come from the project's existing font policy, `pdf_fonts` and `pdf_font_paths`, through `gd-typst`, in regular, bold, italic and bold-italic. With `pdf_fonts = "embedded"` the default is the Typst compiler's embedded serif face, so the drawing needs no system font. We shape with `rustybuzz` and read outlines with `ttf-parser`. In the default build both are already compiled into Guidedog as dependencies of the embedded Typst compiler; we declare them once in `[workspace.dependencies]`, pinned to the versions Typst uses, with a comment giving that reason, as the workspace rules require. `gd-engrave` depends on them directly, so it works without Typst; if Typst later changes its shaper, they remain our own dependencies. Without `gd-typst`'s `embedded` feature, `format: png` is refused with a warning, as `typst-diagram` is, and text faces come only from `pdf_font_paths` and the system; a score with text and no face is refused (`engrave.font`), and a score without text still draws. A cluster that the first face cannot show falls back to the next face; a cluster no face can show is drawn as the face's missing-glyph box, with a warning naming the character.

3.10. Drawing

Display list. The engraving ends as a display list per system: glyphs, lines (staff lines, stems, ledger lines, bar lines, hairpins, hyphens, extenders), four-point polygons (beams and multi-measure rest bars), filled double curves (ties and slurs) and text runs. Each mark carries a *role* (`staff`, `notehead`, `stem`, `beam`, `accidental`, `slur`, `lyric` and so on), used only for classes in the SVG and in the collision audit. Geometry is in points from here on: one staff space is `staff_size / 4` points.
SVG. `write_svg` writes one self-contained document. Its `width` and `height` are in points and its `viewBox` is in points, so a drawing has a real size. Each glyph used is defined once in `<defs>` as a path and placed with `<use>`; text runs become paths too, so the drawing needs no font in the browser. Systems, staves and

measures are groups with classes (`system`, `staff`) and a `data-measure` number on each measure group, for later linking and highlighting. Every mark is filled or stroked with `currentColor`, and the root sets `color: #000`. With `adaptive`, a stylesheet adds `color-scheme: light dark` and, under `@media screen and (prefers-color-scheme: dark)`, a light ink. Only the HTML edition is adaptive; the PDF and PNG editions carry no stylesheet, so they do not depend on how a renderer treats media queries. The document has `role="img"`, a `<title>` from `alt`, and a `<desc>`. `gd-link` words the engraving's `Summary` in the page's language through the UI catalog, for example "Music for 2 staves. Piano. Key signature G major. Time signature 3/4. Measures 1-8. With lyrics." That description fills `<desc>`, and also the alternative text when `alt` is absent; we never use the LilyPond source as alternative text, since it does not describe what a reader sees. Numbers are written with at most three decimals, trailing zeros removed.

Determinism. The same source, options, fonts and Guidedog version give the same SVG bytes on every platform. Arithmetic is IEEE double precision, whose basic operations and square root are exactly rounded everywhere. The elementary functions the engraver needs, $\log_2$ and $\arctan$, come from a small in-house module with fixed polynomial approximations rather than the platform's math library, whose last bits differ between systems. Ties between equal candidate scores are broken by a fixed order, and no hash map iteration order reaches the output. PNG bytes are not promised to be identical across platforms; the rasterizer's antialiasing may differ.

PNG. `format: png` rasterizes the non-adaptive SVG at 144 pixels per inch with the embedded Typst renderer, as `typst-diagram` does, within the same pixel limit.

HTML. The HTML builder engraves at the theme's measure. The stock theme's `--gd-measure` is 47rem, which is 752 CSS pixels or 564 points; a project whose theme or stylesheet changes the column states its measure with `lilypond_html_width`. The builder writes the image inside a figure when there is a caption, marked adaptive through the existing `graphviz-adaptive` class (`color-scheme: inherit`), with or without a caption, so the drawing follows the reader's chosen theme instead of sitting on a white panel.

Phones need more than scaling: on a 360-pixel screen, an 18-point staff scaled down would be about eight points high, with noteheads three pixels tall. The HTML builder therefore engraves a second, *narrow edition* at a 225-point measure (300 CSS pixels), with the staff size reduced to at most 14 points, and writes a `<picture>` whose `<source media="(max-width: 40rem)">` selects it. The narrow edition is broken into its own systems; it is not the wide drawing scaled. It is omitted when the wide drawing is already no wider than 225 points, and also when it would exceed the system limit, in which case narrow screens scale the wide drawing. An explicit `width` applies to the wide edition only. The theme keeps narrow-screen figures with a `lilypond` image from scrolling sideways, unlike graphs.

PDF. The PDF builder engraves at the text width of the template in use. The margins live in the theme's Typst templates and differ between them (31 millimetres a side in `book.typ`, 30 in `article.typ`, 25 in `discussion.typ`), so each built-in template declares its horizontal margin in a leading comment, `// guidedog: margin-x = 31mm`. The declaration follows the initial shared-theme import so that the existing self-contained template assembler can remove that import without removing the margin contract. The builder subtracts twice that margin from the width of `pdf_paper_size`: 148 millimetres for a book on A4. A copied or custom template keeps or writes the same declaration; without one, the builder assumes 25 millimetres, and `lilypond_pdf_width` overrides the result. The drawing is placed at its natural size, so an 18-point staff is 18 points on paper. If it is nevertheless wider than the column, the writer scales it down to the column with Typst's `layout` and the build warns, so a wrong declaration never pushes music into the margin. A drawing with one system is one image. From phase 2, a drawing with several systems is written as one image per system inside a breakable block, so a long example can continue on the next page between systems; the caption follows the last system and numbering is unchanged. In phase 1, a drawing taller than the text height is refused with a hint to split the example.

3.11. Diagnostics

Every error and warning becomes a `gd_report::Report` placed in the document that holds the directive, or in the included `.ly` file, with an exact line, column and caret (see "Crates and their interfaces"). A drawing that fails is not drawn: its source is shown as code with a warning, as Graphviz and Typst diagrams do, and `-W` stops publication. In the examples below, the directive body is indented by three spaces.

```
-- UNSUPPORTED LILYPOND ---------------------------------- theory/phrasing.rst:14:4

Guidedog engraves a fixed house style and does not run overrides.

14|    \override NoteHead.color = #red
       ^^^^^^^^^
Hint: Remove the override. Guidedog reads the part of LilyPond that describes the
music, not the part that programs LilyPond. The source is shown as code.
```

```
-- UNKNOWN COMMAND --------------------------------- theory/articulation.rst:9:6

LilyPond has no command named `\stacato`.

9|    c4\stacato d e f
        ^^^^^^^^
Hint: Correct the spelling. Did you mean `\staccato`?
```

```
-- BAR CHECK FAILED -------------------------------------- theory/scales.rst:22:21

The bar check falls one quarter before the end of measure 2.

22|    c4 d e f | g a b | c1
                          ^
Hint: Check the durations in measure 2: a 4/4 measure holds four quarters, and
these hold three. Engraving continues.
```

The stock codes are `lilypond.syntax`, `lilypond.empty`, `lilypond.unknown-command`, `lilypond.unsupported`, `lilypond.pitch`, `lilypond.duration`, `lilypond.bar-check` (a warning), `lilypond.lyrics` (a warning), `lilypond.include`, `lilypond.limit`, `score.invariant`, `score.time-overflow`, `engrave.overfull` (a warning: one measure is wider than the measure and overflows it), `engrave.font` and `engrave.limit`. The build's warning for a drawing not drawn is `build.lilypond` with the title "MUSIC NOT ENGRAVED". Suggestions use `gd_report::suggest` over the command table and the note names of the current language.

3.12. Limits

Every bound is explicit, returns an error, and never panics or overflows the stack. The parser is recursive descent with a depth counter; variable expansion and (phase 3) `\repeat unfold` are charged to the event limit as they expand.

Table 30 Default limits

Bound	Default
Source, after includes	256 KiB
Nesting depth (braces, << >>, tuplets, variables)	64
Included files	32
Events (notes, rests, chords) after expansion	20,000
Expanded reader allocation allowance	64 MiB, including copied text and notation state
Staves; voices per staff; notes per chord	32; 4 simultaneous voices per staff (64 voice records); 16
Pitch range	octaves -3 to 6 in octave marks (about 25 ledger lines at most)
Measures per line; systems per drawing	128; 64
Drawing height	20,000 points
PNG size	the pixel limit of `typst-diagram`
Ratio numerators and denominators	checked `i64` arithmetic, `i128` comparison
Generated SVG and engraver buffers	charged through `reserve` to the build's memory ledger

The reader and engraver also each have a default work limit of 8,000,000 operations. Token visits, expanded syntax nodes, spacing-candidate gaps, collision comparisons and candidate-curve samples consume work before they run. Exhausting this allowance is a limit diagnostic, including on malformed or adversarial input. A source limit alone does not bound quadratic layout work tightly enough for an untrusted build. The work of each stage is polynomial in these bounds: columns and spacing are $O(m \log m)$ per line, breaking is $O(nw\, m \log m)$ with $w \leq 128$, and beam and slur choices try a fixed number of candidates each. The reader reserves its conservative expanded-allocation allowance before interpretation and 256 bytes per source byte before parsing each source file. Copied strings and state, expanded nodes, spanners and lyric candidates consume the expanded allowance before allocation. The adapter owns a transient ledger for each drawing and releases its reservations after asset publication. Engraver buffers grow with `try_reserve` after `reserve` agrees, so a refused reservation is an error, not an abort.

3.13. Document model and conversions

Table 31 Source and output mapping

Source	Semantic representation	HTML and PDF
RST or MyST `lilypond`	`CodeBlock` with language `lilypond` and custom `lilypond=1`, `lilypond-source`, `lilypond-line`, `lilypond-indent` and, for a file, `lilypond-file`	Engraved image
`caption`	Parsed caption child and caption attribute	Figure caption and numbering
`name` and `class`	Identifier and class attributes	On the image or figure
`alt`, `align`, `format`, `staff-size`, `width`, `justify`	Custom directive attributes	As specified above
Engraved result (after the rewrite)	`Image` with class `lilypond-music` and custom `music-fit`; custom `narrow-src` for the narrow edition; `raster-density` for 144-dpi PNG; custom `music-parts` listing system images for PDF	`<picture>` with `<img>` in HTML; one or several images in PDF
`code-block:: lilypond` or ```` ```lilypond ````	Unmarked code block	Source example

Text output keeps the source and caption without engraving; in a trusted build, a file argument's source replaces its path, and where the file cannot be read a sentence names it instead. The existing reStructuredText writer reconstructs the directive and its options from either RST or MyST input, keeping a file argument as a file argument so that `\include` paths keep their base folder. Read-write-read tests compare notation and options while allowing the new source locations to change. We never reconstruct LilyPond from a drawing. The directive is a Guidedog extension; we do not claim compatibility with `sphinxcontrib-lilypond`, whose `lily` directive and role run the LilyPond program with a configurable preamble. The environment schema changes, so cached documents are parsed again.

3.14. Caching

Engraving is a pure function of the source, the included files, the directive's options, the builder's measure, the text faces and the Guidedog version. We record included files as content dependencies, including failed lookups. As for Typst diagrams, the asset name contains the digest of the output bytes, and an existing asset with the same bytes keeps its modification time, so an unchanged example does not invalidate a cached PDF. Engraving a typical example takes milliseconds, so we engrave again whenever a page is rewritten; a cache keyed by the inputs' digest is a later optimization.

4. Rationale and alternatives

Running LilyPond. LilyPond engraves beautifully and reads its whole language. But it would make an external, GPL-licensed program and its Guile runtime a dependency of a single-executable tool, cost about a second per example, make output depend on the installed version, and, since LilyPond 2.24 dropped its safe

mode, execute arbitrary code from any `.ly` file. It could never run in an untrusted build. We keep LilyPond's language and documented behavior as the specification, and leave an opt-in external engine as an open question.

Linking Verovio. Verovio is a mature LGPL engraving library in C++ that reads MEI, MusicXML, Humdrum, ABC and Plaine & Easie. It reads no LilyPond, so we would still write the reader and translate to MEI. We would add a second C++ build to five platforms beside Graphviz, lose source positions across the translation, and depend on C++ code we cannot change to fix house style. The project's rule is Rust first; music engraving is large, but its algorithms are published and bounded, and owning them lets us integrate with the measure, the theme and the diagnostics.

Drawing through Typst. We could write the display list as Typst code and let the embedded compiler export it. That would give text shaping for free, but the engraver needs text widths before it can space, so it must shape text anyway. Writing SVG directly gives byte-stable output, semantic groups, `currentColor` for the theme, and no compilation per example. We still use the Typst renderer for PNG and the Typst writer for PDF placement.

A generated glyph table. `cargo xtask` could turn the vendored font into a Rust table, so nothing is parsed at run time. But the table would be a modified version of the font under the Open Font License, with renaming obligations, and parsing once per build costs milliseconds. We embed the unmodified files and build the tables explicitly in `load_fonts`.

Spacing rules. Linear spacing in duration wastes space on long notes; square-root rules and the logarithmic rule differ mainly in the ratio of a whole to a quarter. We choose Gourlay's logarithmic rule with LilyPond's function and defaults, so LilyPond users see familiar proportions, and one parameter pair controls it.

Exact solving and optimal breaking. The breakpoint solver gives the exact energy minimizer for adjacent rods at the cost of a sort. A general quadratic program would also handle spanning rods exactly, but adds an iterative solver for a case that conservative conversion handles well. Greedy breaking is simpler; it produces a loose line before a long measure that optimal breaking avoids. Hegazy and Gourlay applied Knuth-Plass breaking to music, and LilyPond has used optimal breaking since its early versions.

Scoring candidates. Beams, slurs and ties have many local rules that interact. Writing them as costs over a small candidate set makes each rule one term, makes the choice explainable, and turns a new rule into a new term rather than a new branch.

A model between reader and engraver. Engraving straight from LilyPond syntax would be shorter at first. A separate score model lets us test interpretation and engraving separately, gives the engraver exact time and spelled pitches only, and leaves room for MusicXML or ABC readers without touching the engraver.

5. Security and operational considerations

The engraver executes nothing the author wrote. It reads no file except through `\include` and the directive's file argument, opens no network connection, and runs no Typst code that the project wrote. Its work and memory are bounded by the limits above, and its buffers and output are charged to the build's memory ledger.

We therefore allow music in `--untrusted` builds, as Graphviz graphs that name no file are allowed. Under `--untrusted`, `\include` and file arguments are refused with a warning and the source is shown as code. In trusted builds, an include resolves beside the including file, then in the project's source folder and its `include_roots`, as RST includes are confined (GDS 0004). It is checked by canonical path, and symbolic links that leave those folders are refused. LilyPond's own search order depends on its `relative-includes` setting; we document ours instead of claiming to match every configuration. The language files (`english.ly` and the others) are not read: they select the note-name language.

Text fonts follow the existing policy; `pdf_fonts = "embedded"` makes drawings independent of the machine. The embedded Bravura files are checked against their digests at build time. A fuzzing target for `load_score` and property tests for the engraver are part of the acceptance evidence, since the reader faces untrusted text. Failures never fall back to an earlier drawing: an asset is published only from a successful engraving, under a name derived from its bytes.

6. Backwards compatibility

We add a directive, three configuration settings and three internal crates. The built-in Typst templates gain a margin declaration comment; copied templates without one fall back as described in "PDF". Existing code blocks, including `lilypond` code blocks, keep their meaning. Projects that use the directive need no installed program. The executable grows by the embedded font and metadata (about 2 MB) and the engraver's code; `rustybuzz` and `ttf-parser` are already linked in the default build. The release's notices (`NOTICE`, `--licenses` and the archive's license files) gain Bravura's copyright and the SIL Open Font License. The theme's stylesheet gains a narrow-screen rule for `lilypond` figures, which projects with a copied theme can add. The environment schema changes.

7. Acceptance criteria and implementation status

All three notation phases are implemented in the native reader, score model, engraver and project adapters. The qualification record is Native music implementation review, 1 November 2026. The complete local checks below pass, including all translated books and workspace validation; the release matrices enforce the same corpus on each release platform.

Phase 1 is one staff with one voice: the phase 1 profile, lyrics of one verse, beams, ties, slurs, tuplets, articulations, dynamics and hairpins, line breaking, SVG and PNG, the HTML and PDF builders with single images, diagnostics with exact positions, and limits. The score model includes grace moments, staff groups and voices from the start, so later phases do not change it. Phase 2 adds multiple staves and staff groups, four voices per staff, several verses, phrasing slurs, the narrow HTML edition and per-system PDF images. Phase 3 adds grace notes, repeats with voltas and alternatives, unfolded repeats, tempo and rehearsal marks, and chord names. Each phase is complete when this evidence exists for its constructs:

- Reader tests for every construct in the profile table, in RST and in MyST, including each refusal's report with its exact position, and every relative-octave and transposition example of LilyPond's Notation Reference that lies in the profile.
- Property tests: no input to `load_score` panics or exceeds its limits (also run under a fuzzer); ideal gaps never decrease with duration; the solver's gaps satisfy every original rod and sum to the measure; on random small scores, the breaker's result equals exhaustive search.
- A collision audit run on every engraved test score. It checks marks by role against a table of allowed contacts (stems on noteheads, ledger lines through noteheads, beams on stems, bar lines and staff lines on everything they cross, slurs across staff lines) and reports any other overlap beyond a tolerance of 0.05 staff space. It also checks that exposed primary beam ends avoid forbidden quants and no tie apex touches a staff line. Secondary and fractional ends share the chosen slope and beam spacing; their quant wedges contribute to the placement cost instead of becoming independent hard constraints.
- A reference corpus of at least forty examples covering the house style (Gould's chapters on spacing, beams, accidentals, ties and slurs, lyrics and multiple voices), stored as golden SVGs. CI compares them byte for byte on Linux and macOS; the release qualification compares them on all five release platforms. PNG output is compared by pixel difference within a tolerance.
- A visual review of the corpus beside LilyPond 2.24 output, recorded as an evidence file under `docs/gds/evidence`, noting each deliberate difference.
- Project tests for captions, labels and numbering, `format: png`, file arguments, includes and their invalidation, `--untrusted`, localized descriptions, and real HTML and PDF builds (from phase 2, with the narrow edition and multi-system examples).
- A manual chapter on music, with its translations, and an example page in the manual's own build.
- `cargo build`, `cargo test`, `cargo clippy --all-targets`, `cargo fmt --check` and the source limit checker pass.

8. Open questions

- Should an inline `:lilypond:` role draw a short fragment in a sentence, sized to the text's line height?
- Should trusted projects be able to opt in to the external LilyPond program, for scores outside the profile, with the program's output treated like any other generated image?
- Should the drawing carry a MIDI rendering or audio, and should the `data-measure` groups be scripted for playback highlighting?

- Should projects be able to choose another SMuFL font (Leland or Petaluma, both under the Open Font License), or should the house style keep one?
- Should the HTML edition re-break in the browser at any width, instead of offering two editions?
- Should `guidedog migrate` convert `sphinxcontrib-lilypond` directives whose source lies in the profile?

9. Review history

2026-11-01, initial draft. It chose an in-house engraver over the LilyPond program and over Verovio, a score model between reader and engraver, direct SVG with glyph outlines, and per-builder measures.
2026-11-01, first review, by an independent reviewer and the author, with the LilyPond sources checked.

- Critique: the spring solver's optimality argument had a sign error in its multiplier, and the derived `Ord` on a numerator-denominator pair would order 1/2 below 1/3. Revision: correct the conditions to $F + \mu_i$; implement `Ord` by cross-multiplication.
- Critique: the line-breaking rules could leave no solution (a measure wider than the line was neither a candidate nor allowed to overflow), squared negative penalties made double bars the worst breaks, and compression badness was bounded. Revision: overfull single-measure and forced lines are always candidates with a fixed badness; use Knuth and Plass's $-p^2$ for negative penalties; make compression badness unbounded.
- Critique: the spacing function was attributed to LilyPond but differed from it for short notes and in the cap on the common shortest duration; rods were only between adjacent columns, which misses one staff's content across another staff's columns. Revision: adopt `get_duration_space` and `base-shortest-duration` exactly as LilyPond's source has them; derive rods per staff and convert spanning rods in a fixed order.
- Critique: the draft's bow construction was its own rather than LilyPond's, and the reviewer believed LilyPond had no `bezier-bow` code. Revision: the source shows `lily/bezier-bow.cc` with `slur_height` and `get_slur_indent_height`; adopt its height, its smooth indent and its control-point convention, under which the apex is three quarters of the stated height.
- Critique: the include callback could not resolve nested relative includes or report resolved names, the error type could not place spans in included files, body offsets would not map to RST columns, the font table was a lazy global, font bytes were copied per call, and descriptions were English only. Revision: a `FnMut` include callback with the including file and a resolved name, called by the parser; `Failed` keeps the files read; the directive records its body's line and indentation; `load_fonts` builds explicit tables over borrowed bytes with bold and italic faces; the engraver returns a `Summary` that `gd-link` words through the UI catalog.
- Critique: phase 1 held several engines' worth of work, many behaviors were unspecified (multi-measure rests, key cancellation, clef changes, two-voice shifts, rests in two voices, lyrics on rests and leftover syllables, tuplet brackets, bar numbers, empty input), and measures were hard-coded for the stock theme and A4. Revision: three phases with a model complete from the start; one rule for each listed behavior; measures from the theme's column and from a margin each Typst template declares, with project overrides and a scale-to-fit safety net. Checking the templates also showed that the PDF margins live in `book.typ`, `article.typ` and `discussion.typ`, not in the Typst writer, and differ between them.
- Critique: the dark-mode stylesheet relied on renderers ignoring a media query, and five-platform byte equality could not run in CI. Revision: only the HTML edition carries the stylesheet, through the existing adaptive class; CI compares SVGs on Linux and macOS and the release job on five platforms, and PNGs by pixel tolerance.

2026-11-01, implementation review.

- Critique: requiring every line before a forced break to start at the preceding forced break eliminates valid intermediate lines. Revision: a candidate may not cross a forced break, but may start at any eligible break after the preceding one.
- Critique: the 128-measure search window makes the global-optimum claim conditional, and polynomial work over 20,000 events can still be excessive. Revision: state the admissible window precisely and charge parser and engraver operations to separate finite work budgets.

- Critique: the common diagnostic API couples the engraver to the LilyPond reader through its source-file type. Revision: keep spans and diagnostic data in the dependency-free score crate and report conversion in the reader adapter.
- Critique: a constant body indent cannot recover exact source positions through tabs, differently indented lines or nested includes. Revision: preserve a per-line source mapping and include identities; test multibyte text, included errors and nonuniform indentation.
- Critique: a three-phase proposal could be marked complete after only basic note rendering, and unsupported commands could disappear during incremental development. Revision: distinguish implemented and planned commands, reject planned constructs explicitly and require every phase before commitment.
- Critique: a large native engraver needs reviewable evidence beyond successful compilation. Revision: require recorded corpus checks, bounded-work failures, real HTML/PDF output and a final gap review, with an explicit status for every acceptance item.

Implementation integration review, 1 November 2026.

- File arguments must be refused before they are opened in an untrusted build, so the markup reader retains their paths and the trusted link pass performs the confined read.
- The existing conversion renderer writes RST from both RST and MyST input; the music directive uses that path and retains file arguments, options and source text.
- Text face data needs an explicit collection index to preserve style selection and avoid reading all collection faces as regular text.
- Built-in templates start with an import that the standalone assembler removes; the margin declaration therefore follows that import rather than replacing its first line.

9.1. Documentation plan

The manual gains a music-notation chapter under Topics, next to diagrams, with a live engraved example and LilyPond source beside it. It explains pitches and relative octaves, durations, chords, rests, voices, staff groups, lyrics, markings, grace notes and repeats for the implemented profile. The same chapter (`topics/music.rst`) covers file-based scores, nested includes, SVG and PNG, builder measures, narrow screens, multi-page PDFs, trust, limits and incremental builds. The directive reference lists every option and fallback; configuration reference lists the three settings and defaults. The built-in extension inventory, crate map, all three new crate READMEs, project README, changelog and release/license documentation describe the feature and its boundaries. The manual links to LilyPond's notation reference and SMuFL; references distinguish accepted notation from engraver programming. All six manual translations are refreshed and translated, with zero fuzzy or untranslated entries for the added material. The new chapter, examples and translated HTML/PDF books are built as part of qualification. An evidence record documents design review, implementation review, corpus inspection, deliberate engraving differences and every remaining gap before the state can become committed.

10. References

- The Guidedog Discussion Process, Guidedoc: the conversion engine, Memory budgets and what a build may read, Embedded Typst diagrams
- Elaine Gould, *Behind Bars: The Definitive Guide to Music Notation*, Faber Music, 2011.
- Ted Ross, *The Art of Music Engraving and Processing*, Hansen Books, 1970.
- Donald E. Knuth and Michael F. Plass, "Breaking Paragraphs into Lines", *Software: Practice and Experience* 11(11), 1981.
- John S. Gourlay, "Spacing a Line of Music", Technical Report OSU-CISRC-10/87-TR35, Ohio State University, 1987.
- Wael A. Hegazy and John S. Gourlay, "Optimal Line Breaking in Music", technical report, Ohio State University, 1987.
- Lippold Haken and Dorothea Blostein, "A New Algorithm for Horizontal Spacing of Printed Music", *Proceedings of the International Computer Music Conference*, Banff, 1995.
- Kai Renz, *Algorithms and Data Structures for a Music Notation System based on GUIDO Music Notation*, doctoral thesis, Technische Universitaet Darmstadt, 2002.

- Han-Wen Nienhuys and Jan Nieuwenhuizen, “LilyPond, a System for Automated Music Engraving”, *Proceedings of the XIV Colloquium on Musical Informatics*, Firenze, 2003.
- LilyPond sources: `lily/spacing-options.cc` (`get_duration_space`), `lily/spacing-spanner.cc` (common shortest duration), `lily/bezier-bow.cc` (bow shape), `scm/define-grobs.scm` (defaults) and `scm/time-signature-settings.scm` (beam groups), at github.com/lilypond/lilypond.
- LilyPond writing pitches and horizontal spacing overview
- LilyPond vertical collision avoidance
- LilyPond slur interface and debugging scoring algorithms
- CVE-2020-17354
- SMuFL specification and Bravura
- Verovio
- sphinxcontrib-lilypond

Lifecycle event

```
2026-11-01: prediscussion -> discussion. Review the native engraving design,
source mapping, bounded work and complete three-phase acceptance contract.
```

Lifecycle event

```
2026-11-01: discussion -> accepted. Implement the reviewed three-crate design
and qualify each phase against the explicit profile and documentation
contract.
```

10.1. Resource review

Event and operation limits alone do not bound repeated text or cloned notation state. The reader therefore accounts for these copies before expansion, with a separate allocation allowance and an optional conservative host reservation. A regression expands a large text attachment under a small allowance and requires a placed limit diagnostic. Transient music reservations belong to the directive, so a page containing many small examples does not retain every example’s peak allocation until the page ends.

10.2. Beam quantization review

A hard quant constraint on every secondary endpoint is overconstrained for a fixed slope and fixed beam spacing. A dense group can have no admissible candidate inside the declared quarter-space search even though it is ordinary notation. We therefore require allowed exposed primary ends, score inner-end wedges with a named quant weight, and include notehead and accidental collisions for every actual segment in the candidate cost. Scoring and SVG drawing share the same fractional segment endpoints. The dense-duration corpus and a primary-end geometry regression check this rule.

10.3. Allocation review

The engraving entry point reserves bounded stage scratch and the retained worker capacities before preparing columns. This includes the largest temporary shaper expansion, using Rustybuzz’s 64-glyph-per-input bound, and conservative font, column, band and snapshot allowances. Outline strings consult the caller’s ledger before growth, stop at 65,536 bytes per glyph and consume work by serialized path length before accumulating in a text run. Text glyph ink is bounded to sixteen ems on either axis before it enters quarter-space bands. Generated SVG grouping reserves its tree and index buffers before insertion and rejects undefined glyph references transactionally. The positioned-mark limit applies across all systems in a drawing, and impossible system-window counts are refused before column allocation. Measure expansion consumes the reader’s allocation allowance as well as its event-derived measure-count bound. These rules close paths that event and source limits alone did not cover.

10.4. Final geometry review

A role-ink corpus audit found cautionary parentheses touching ledger lines and a flat signature touching its clef. We now place parentheses from actual glyph bounds, reserve their rods and give prefatory clefs explicit clearance. Accidental state is updated atomically for each moment; conflicting simultaneous alterations leave that pitch ambiguous until a later explicit note resolves it. Accidental columns use quarter-space bands refined by the SMuFL cut-outs. Spacing retains the last occupied column separately for each vertical

band, including ledger and rest footprints, so an intervening unrelated column cannot erase a spanning clearance.
Slur candidates use quarter-space attachment offsets, actual stem tips when appropriate, sampled minimum height adjustment and named costs for attachments, slope, added height, clearance variance and ties. Attached curves remain fixed while the outside-mark skyline places text and annotations around them; translating a whole slur during outside placement would detach its ends. Tie apexes move to a staff space when they approach a staff line. Lyrics use shaped word spacing, actual text bounds and continuation anchors, including timed lyric moments. System frames include all horizontal ink, and PNG stores physical resolution so its PDF size matches SVG.
The corpus audit rejects disallowed contacts using actual 288-dpi alpha masks after a bounding-box rejection. It permits the component contacts within signatures, connected beams, brackets, dynamic strings and composite measure rests in addition to the contacts listed above. Only ends with no adjoining primary segment are exposed beam ends. The exhaustive tests compare spring solutions with every active set and line breaking with every partition of small generated scores. The deterministic fuzz target reports accepted and engraved inputs as well as total cases.

10.5. Post-implementation review, 1 November 2026

A second review against LilyPond 2.24 and Gould's rules expanded the profile and closed remaining semantic, geometric and builder gaps:

- `gd-score`, `gd-lilypond` and `gd-engrave` now represent and render a bounded LilyPond 2.24 `\markup` tree (lists, aligned columns, styles, small caps, sub/superscripts, relative and absolute sizes, enclosures, offsets, spacing, music symbols and markup variables) across scripts, `\tempo`, `\mark` and instrument names, while refusing Scheme, layout and embedded-score markup commands explicitly.
- `\alternative` supports `\volta` pass lists as well as sequential endings; `\fine`, `\section`, `\compressEmptyMeasures`, `\compressFullBarRests`, `\numericTimeSignature` and `\defaultTimeSignature` are accepted.
- `\relative` without an explicit pitch uses LilyPond's absolute-octave middle reference (`f`); `\transpose` transposes `\key` tonics with diatonic spelling; `\acciaccatura` and `\appoggiatura` add implicit slurs to their main notes; melismata skip manual beams only when automatic beaming is off; `\lyricsto` resolves voices created later in simultaneous blocks.
- Courtesy clefs and courtesy key/time signatures appear before line-start changes; key changes cancel dropped accidentals before drawing new sharps or flats; back-to-back `:|.` and `.|:` merge into `:..:`; concave and equal-endpoint beams stay strictly horizontal; 3/8 groups across the whole measure; two-voice dots and chords use per-head X positions and voice-two downward dot preferences.
- Multi-system PDF figures emit a single numbered `#figure` around a breakable block of system images, template margin comments are scanned in the leading 256 lines and measured on the paper the book renders (A4 or US letter), directive widths accept standard print and CSS units, narrow HTML `<picture>` editions are omitted when the wide drawing already fits inside 225 points, and external `.ly` source is substituted into trusted text and fallback code blocks.
- `music-check` now audits both the standard 420-point and compact 225-point editions across 58 golden scores at 288 pixels per inch.

Lifecycle event

```
2026-11-01: accepted -> committed. Implement and review all three native
music phases, project integration, complete documentation and translations;
pass local qualification and wire platform corpus gates.
docs/gds/evidence/native-music-20261101.rst; 48-score SVG/PNG and collision
corpus; 12000 fuzz cases; 2065 workspace tests; seven HTML/PDF manual
editions; strict Clippy, formatting, source and license checks.
```

10.6. Follow-up qualification

The mixed-script corpus now includes Chinese and Devanagari lyrics with fixed font bytes. CI and every native release job download the single pinned CJK face before running the corpus; the loader checks its digest again. At both staff sizes the corpus refuses missing-glyph and other engraving warnings. Direct

HTML and PDF conversion names an unengraved file argument through a localized label without opening it; RST retains the original directive.
`Options::scratch` supplies reserve and release callbacks for temporary layout storage. Skyline segment owners retain their reservation while alive, including the overlap of old and replacement buffers during a merge. Spacing sort buffers, local rods and break-choice tables release their reservations on both success and error. The project adapter connects them to `Ledger::charge` and `Ledger::discharge`; that refund operation avoids counting every intermediate merge cumulatively. Retained workspace capacity is accounted for each call because callers can replace the ledger between scores.
System context uses a measure index, independently of its printed number, and voice-column arrangement runs after beam directions are final. The maintainer subsequently extended this follow-up to complete third and fourth voices. The four-voice policy, bounded collision layout and fifth-voice refusal above are required parts of this revision. Qualification includes implicit and named voices, dense seconds and unisons, dots and accidentals, mixed-duration rests, beams, lyrics and ties across systems.
The music chapter must also contain a small, live teaching example from a famous public-domain composition. We use four bars of Beethoven's Joy theme from Symphony No. 9, finale, bars 92-95, in C major and the treble register. We retain the instrumental melody's half-note opening and dotted rhythm, omit orchestration, slurs and dynamics, cite the source score, and explain the pitches and beat counts. The example source is checked by a pitch-and-duration regression and the image corpus. The manual includes copyable RST and MyST forms, a caption and accessible description; every manual translation covers the explanation and new voice limit.

10.7. Four-voice review, 1 November 2026

An independent review of the four-voice revision and its follow-ups found layout defects that the corpus did not exercise:

- Critique: ties and slurs in voices three and four rose into spikes up to a hundred points tall, because the clearance formula divides by a small number near a curve's ends. Revision: ties rise at most 1.5 staff spaces; in polyphony a slur rises at most 2.5 and may cross another voice, as LilyPond's slurs avoid only their own voice.
- Critique: articulations, fingerings and dynamics of two voices on one side printed over each other at the column's x, and a left-shifted fourth voice could stand left of the previous beat. Revision: scripts follow the shifted heads, overlapping single-glyph marks stack outward, and the heads of a column with shifted voices form one unit for spacing.
- Critique: `\voiceOne` moved dynamics above the staff, a voice that began with `<< \\ >>` kept upward stems afterwards, measure rests and tuplet numbers ignored their voice, and identical accidentals of two voices printed twice. Revision: dynamics keep their direction, as LilyPond's `direction-polyphonic-grobs` omit `DynamicLineSpanner`; the voice record keeps its neutral direction; measure rests and tuplets follow the voice; one accidental serves voices that sound the same pitch.
- Critique: the voice limit counted voice records, so five differently named voices failed although at most three sounded at once. Revision: the reader and `check_score` count voices sounding at once, with a bound of sixty-four records per staff; `\shiftOn` and its relatives are refused instead of reported as unknown.
- Critique: conversion renderers each knew the `lilypond` directive's attribute and message, and the music corpus fuzz target needed the 25 MB CJK download. Revision: one engine helper names the unengraved score file; fuzzing runs without the CJK face, which CI caches by the font list's digest.

10.8. Open-issue follow-up, 1 November 2026

The four-voice review left four items open, now closed:

- Critique: two voices' dynamics or hairpins at one moment printed over each other, because a dynamic is drawn glyph by glyph and its letters could not be told from another voice's sign. Revision: the drawing records multi-glyph signs, and vertical placement stacks whole signs outward.
- Critique: `\shiftOn` and its relatives were refused. Revision: events carry an explicit column shift level, ordered and offset as in LilyPond's note-collision code; default four-voice layout is unchanged.
- Critique: about sixty bars of beamed eighths exhausted the engraving work limit, because each line candidate scanned every rod, the breaker kept trying starts after a line overflowed, and each beam candidate tested every notehead in the system. Revision: rods are indexed by their first gap, the search

stops once no earlier start can fit unless the line ends at a forced break, and a beam tests only the notes beneath it; work now grows linearly, and the output is unchanged. The breaking property test still matches a brute-force optimum.
- Critique: catalogs kept obsolete messages. Revision: the manual and site catalogs were updated with `--no-obsolete`; their live translations are unchanged.

10.9. Gap follow-up, 2 November 2026

The maintainer asked to close three gaps found while writing the songbook task:

- Critique: metronome marks printed their beat unit as text, such as `1/4 = 120`. Revision: the unit is a SMuFL metronome note with its dots, scaled beside the text, and the rate follows in parentheses after a label, as LilyPond prints `\tempo "Allegro" 4 = 120`.
- Critique: `\breathe` was outside the profile, though vocal music needs it. Revision: a breath mark prints as a comma just above the staff after its note and takes its own room in the spacing; without a preceding note it is refused with a placed diagnostic.
- Critique: string numbers chose tablature strings but did not print on standard staves; GDS 0016 records their circled digits.