

LilyPond

Das Notensatzprogramm

Das LilyPond-Entwicklerteam

Copyright © 1999–2007 bei den Autoren

The translation of the following copyright notice is provided for courtesy to non-English speakers, but only the notice in English legally counts.

Die Übersetzung der folgenden Lizenzanmerkung ist zur Orientierung für Leser, die nicht Englisch sprechen. Im rechtlichen Sinne ist aber nur die englische Version gültig.

Es ist erlaubt, dieses Dokument unter den Bedingungen der GNU Free Documentation Lizenz (Version 1.1 oder spätere, von der Free Software Foundation publizierte Versionen), zu kopieren, verbreiten und/oder zu verändern. Eine Kopie der Lizenz ist im Abschnitt “GNU Free Documentation License” angefügt.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections. A copy of the license is included in the section entitled “GNU Free Documentation License”.

(For LilyPond version 2.11.31)

Inhaltsverzeichnis

Vorwort	1
1 Einleitung	2
1.1 Notensatz	2
1.2 Automatisierter Notensatz	3
1.3 Welche Symbole?	5
1.4 Musikalische Repräsentation	6
1.5 Beispielanwendung	8
1.6 Über das Handbuch	9
2 Übung	11
2.1 Erste Schritte	11
2.1.1 Eine Quelldatei übersetzen	11
2.1.2 Einfache Notation	12
2.1.3 Arbeiten an Text-Dateien	16
2.1.4 Wie soll man die Übung lesen	17
2.2 Notation auf einem System	17
2.2.1 Relative Notenbezeichnungen	17
2.2.2 Versetzungszeichen und Tonartbezeichnung (Vorzeichen)	18
2.2.3 Bindebögen und Legaotbögen	19
2.2.4 Artikulationszeichen und Lautstärke	20
2.2.5 Automatische und manuelle Balken	22
2.2.6 Zusätzliche rhythmische Befehle	22
2.3 Mehrere Noten auf einmal	23
2.3.1 Musikalische Ausdrücke erklärt	23
2.3.2 Mehrere Notensysteme	25
2.3.3 Klaviersysteme	26
2.3.4 Mehrstimmigkeit in einem System	27
2.3.5 Noten zu Akkorden verbinden	28
2.4 Lieder	28
2.4.1 Setzen von Text	28
2.4.2 Ein Song-Blatt	30
2.5 Letzter Schliff	31
2.5.1 Versionsnummer	31
2.5.2 Titel hinzufügen	31
2.5.3 Absolute Notenbezeichnungen	31
2.5.4 Stücke durch Bezeichner organisieren	33
2.5.5 Nach der Übung	34
2.5.6 Wie soll das Handbuch gelesen werden	34
3 Alles zusammenfügen	35
3.1 Erweiterung der Beispiele	35
3.2 Wie eine LilyPond-Datei funktioniert	38
3.3 Score ist ein einziger musikalischer Ausdruck	39
3.4 Eine Orchesterstimme	41

4	An LilyPond-Projekten arbeiten	44
4.1	Vorschläge, wie LilyPond-Dateien geschrieben werden sollen	44
4.1.1	Allgemeine Vorschläge	44
4.1.2	Das Kopieren von existierender Musik	45
4.1.3	Große Projekte	45
4.2	Tipparbeit sparen durch Bezeichner und Funktionen	46
4.3	Stil-Dateien	48
4.4	Alte Dateien aktualisieren	51
4.5	Fehlersuche (alles auseinander bauen)	51
4.6	Minimalbeispiele	52
5	Die Ausgabe verändern	53
5.1	Verschieben von Objekten	53
5.2	Überlappende Notation in Ordnung bringen	56
5.3	Übliche Optimierungen	56
5.4	Standarddateien	58
5.5	Die Musik auf weniger Seiten zwingen	59
5.6	Fortgeschrittene Optimierungen mit Scheme	60
5.7	Vermeiden von Optimierungen durch langsamere Übersetzung	61
6	Grundlegende Notation	62
6.1	Tonhöhen	62
6.1.1	Normale Tonhöhen	62
6.1.2	Versetzungszeichen	63
6.1.3	Warnungsversetzungszeichen	64
6.1.4	Mikrotöne	64
6.1.5	Notenbezeichnungen in anderen Sprachen	65
6.1.6	Relative Oktaven	65
6.1.7	Oktavenüberprüfung	67
6.1.8	Transposition	67
6.1.9	Pausen	68
6.1.10	Überspringen von Zeichen	69
6.2	Rhythmus	70
6.2.1	Tondauern	70
6.2.2	Punktierung	70
6.2.3	Andere rhythmische Aufteilungen	71
6.2.4	Tondauern skalieren	73
6.2.5	Taktüberprüfung	74
6.2.6	Taktnummerüberprüfung	74
6.2.7	Automatische Aufteilung von Noten	74
6.3	Mehrstimmigkeit	75
6.3.1	Akkorde	75
6.3.2	Hälse	75
6.3.3	Einfache Mehrstimmigkeit	76
6.3.4	Stimmen explizit beginnen	77
6.3.5	Auflösung von Zusammenstößen	79
6.4	Notation innerhalb von Systemen	81
6.4.1	Notenschlüssel	81
6.4.2	Tonartbezeichnung	83
6.4.3	Taktangabe	84
6.4.4	Auftakte	85
6.4.5	Taktlinien	86
6.4.6	Musik ohne Metrum	88

6.4.7	Klammern am Systemanfang	88
6.4.8	Das Notensystem	90
6.4.9	Musik parallel notieren	91
6.5	Noten verbinden	92
6.5.1	Bindebögen	92
6.5.2	Legatobögen	94
6.5.3	Phrasierungsbögen	95
6.5.4	Laissez vibrer-Bögen	96
6.5.5	Automatische Balken	96
6.5.6	Manuelle Balken	97
6.5.7	Verzierungen	98
6.6	Ausdrucksbezeichnungen	101
6.6.1	Artikulationszeichen	101
6.6.2	Fingersatzanweisungen	103
6.6.3	Dynamik	104
6.6.4	Atemzeichen	107
6.6.5	Triller	107
6.6.6	Glissando	108
6.6.7	Arpeggio	109
6.6.8	Glissando zu unbestimmter Tonhöhe	110
6.7	Wiederholungszeichen	110
6.7.1	Wiederholungstypen	110
6.7.2	Die Syntax der Wiederholungen	110
6.7.3	Wiederholungen und MIDI	112
6.7.4	Manuelle Wiederholungsbefehle	113
6.7.5	Tremolo-Wiederholung	114
6.7.6	Tremolo-Unterteilung	115
6.7.7	Taktwiederholungen	115
7	Instrumentenspezifische Notation	117
7.1	Notation für Klavier	117
7.1.1	Automatische Notensystemwechsel	117
7.1.2	Manuelle Notensystemwechsel	117
7.1.3	Pedalbezeichnungen	117
7.1.4	Stimmführungslinien	117
7.1.5	Häse über beide Systeme	117
7.2	Akkordbezeichnungen	117
7.2.1	Einführung in Akkordbezeichnungen	117
7.2.2	Akkord-Modus	117
7.2.3	Akkordbezeichnungen drucken	117
7.3	Notation von Gesang	117
7.3.1	Einfache Lieder setzen	117
7.3.2	Eingabe von Text	117
7.3.3	Hyphens and extenders	117
7.3.4	The Lyrics context	117
7.3.5	Melismata	118
7.3.6	Eine andere Art, den Text einzugeben	118
7.3.7	Flexibilität bei der Positionierung	118
7.3.7.1	Text zu mehreren Noten eines Melismas	118
7.3.7.2	Getrennte Texte	118
7.3.7.3	Die Melodie, die mit einer Textzeile verbunden ist, umschalten	118
7.3.7.4	Specifying melismata within the lyrics	118
7.3.7.5	Text unabhängig von den Noten	118
7.3.8	Textabstände	118

7.3.9	Mehr über Strophen	118
7.3.9.1	Adding stanza numbers	118
7.3.9.2	Lautstärkebezeichnung hinzufügen	118
7.3.9.3	Sängernamen hinzufügen	118
7.3.9.4	Printing stanzas at the end	118
7.3.9.5	Printing stanzas at the end in multiple columns	118
7.3.10	Ambitus	118
7.3.11	Weitere Vokalmusikprobleme	118
7.4	Rhythmische Musik	118
7.4.1	Melodierhythmus anzeigen	119
7.4.2	Schlagzeugnotation	119
7.4.3	Schlagzeugsysteme	119
7.4.4	Geisternoten	119
7.5	Gitarre	119
7.5.1	Seitennummerbezeichnung	119
7.5.2	Grundlagen der Tabulatur	119
7.5.3	Nicht-Gitarren-Tabulaturen	119
7.5.4	Banjo-Tabulaturen	119
7.5.5	Bund-Diagramme	119
7.5.6	Fingersatz der rechten Hand	119
7.5.7	Weiter Gitarrenprobleme	119
7.6	Dudelsack	119
7.6.1	Dudelsack-Definitionen	119
7.6.2	Dudelsack-Beispiele	119
7.7	Notation von alter Musik	119
7.7.1	Notenköpfe der alten Musik	120
7.7.2	Versetzungszeichen der alten Musik	120
7.7.3	Pausen der alten Musik	120
7.7.4	Schlüssel der alten Musik	120
7.7.5	Fähnchen der alten Musik	120
7.7.6	Taktangaben der alten Musik	120
7.7.7	Vorzeichen der alten Musik	120
7.7.8	Custodes	120
7.7.9	Divisiones	120
7.7.10	Ligaturen	120
7.7.10.1	Weißer Mensuralligaturen	120
7.7.10.2	Ligaturen der gregorianischen Quadratnotation	120
7.7.11	Gregorianische Gesangs-Kontexte	120
7.7.12	Mensural-Kontexte	120
7.7.13	Musica ficta-Versetzungszeichen	120
7.7.14	Generalbass	120
7.8	Notation für verschiedene Instrumente	121
7.8.1	Flageolet (Streicher)	121
8	Fortgeschrittene Notationstechniken	122
8.1	Text	122
8.1.1	Textarten	122
8.1.2	Text und Linien	122
8.1.3	Text mit Verbindungslinien	122
8.1.4	Textartige Zeichen	122
8.1.5	Textbeschriftung	122
8.1.6	Geschachtelte Systeme	122
8.1.7	Page wrapping text	122
8.1.8	Überblick über Textbeschriftungsbefehle	122

8.1.9	Overview of text markup list commands	130
8.1.10	Auswahl der Schriftart	130
8.1.11	Neue Lautstärkezeichen	130
8.2	Orchesterstimmen vorbereiten	131
8.2.1	Mehrtaktige Pausen	131
8.2.2	Metronombezeichnung	131
8.2.3	Übungszeichen	131
8.2.4	Taktnummern	131
8.2.5	Instrumentenbezeichnungen	131
8.2.6	Transposition von Instrumenten	131
8.2.7	Oktavierungsklammern	131
8.2.8	Verschiedene Editionen aus einer Quelldatei	131
8.3	Orchestermusik	131
8.3.1	Automatische Kombination von Stimmen	131
8.3.2	Systeme verstecken	131
8.3.3	Stichnoten	131
8.3.4	Stichnoten formatieren	131
8.3.5	An Kadenzen ausrichten	131
8.4	Moderne Notation	131
8.4.1	Polymetrische Notation	132
8.4.2	Verwaltung der Zeiteinheiten	132
8.4.3	Proportionale Notation (Einleitung)	132
8.4.4	Cluster	132
8.4.5	Besondere Notenköpfe	132
8.4.6	Gespreizte Balken	132
8.4.7	Improvisation	132
8.4.8	Auswahl der Notations-Schriftgröße	132
8.5	Pädagogische Verwendung	132
8.5.1	Erklärungsblasen	132
8.5.2	Ein leeres Notenblatt	132
8.5.3	Unsichtbare Noten	132
8.5.4	Notenköpfe mit besonderen Formen	132
8.5.5	Easy Notation-Notenköpfe	132
8.5.6	Analyseklammern	132
8.5.7	Farbige Objekte	132
8.5.8	Klammern	132
8.5.9	Gitternetzlinien	133
9	Standardeinstellungen verändern	134
9.1	Automatic notation	134
9.1.1	Automatische Versetzungszeichen	134
9.1.2	Einstellung von automatischen Balken	134
9.2	Interpretationsumgebungen	134
9.2.1	Was sind Umgebungen?	134
9.2.2	Umgebungen erstellen	134
9.2.3	Umgebungseigenschaften lokal ändern	134
9.2.4	Umgebungs-Plugins verändern	134
9.2.5	Layouteinstellungen mit Umgebungen	134
9.2.6	Die Standardeinstellungen von Umgebungen ändern	134
9.2.7	Neue Umgebungen definieren	134
9.2.8	Umgebungen aneinander ausrichten	134
9.2.9	Vertical grouping of grobs	134
9.3	The <code>\override</code> command	134
9.3.1	Eine Korrektur konstruieren	134

9.3.2	Zurechtfinden in der Programmreferenz	135
9.3.3	Layout-Schnittstellen	135
9.3.4	Die Grob-Eigenschaften	135
9.3.5	Objekte, die mit der Eingabe verbunden sind	135
9.3.6	Using Scheme code instead of <code>\tweak</code>	135
9.3.7	<code>\set</code> vs. <code>\override</code>	135
9.3.8	Schwierige Korrekturen	135
10	Nichtmusikalische Notation	136
10.1	Quelldateien	136
10.1.1	Die Dateistruktru (Einleitung)	136
10.1.2	Die Dateistruktur	136
10.1.3	Ein einzelner musikalischer Ausdruck	136
10.1.4	Mehrere Partituren in einem Buch	136
10.1.5	Fragmente der Notation extrahieren	136
10.1.6	LilyPond-Dateien einfügen	136
10.1.7	Textkodierung	136
10.2	Titel	136
10.2.1	Titel erstellen	136
10.2.2	Eigene Titel	136
10.2.3	Reference to page numbers	136
10.2.4	Table of contents	136
10.3	MDID-Ausgabe	136
10.3.1	MIDI-Dateien erstellen	136
10.3.2	Der MIDI-Block	137
10.3.3	MIDI-Instrumentenbezeichnungen	137
10.4	LilyPond-Notation anzeigen	137
10.5	Korrigierte Musik überspringen	137
11	Abstände	138
11.1	Papier und Seiten	138
11.1.1	Papiergröße	138
11.1.2	Seitenformatierung	138
11.2	Notenlayout	138
11.2.1	Die Notensystemgröße einstellen	138
11.2.2	Partiturlayout	138
11.3	Abstände anzeigen lassen	138
11.4	Umbrüche	138
11.4.1	Zeilenumbrüche	138
11.4.2	Seitenumbrüche	138
11.4.3	Optimale Seitenumbrüche	138
11.4.4	Optimale Umbrüche zum Blättern	138
11.4.5	Ausdrückliche Umbrüche	138
11.4.6	Eine zusätzliche Stimme für Umbrüche benutzen	138
11.5	Vertikale Abstände	139
11.5.1	Vertikale Abstände innerhalb eines Systemes	139
11.5.2	Vertikale Abstände zwischen Systemen	139
11.5.3	Explizite Positionierung von Systemen	139
11.5.4	Vertikale Abstände mit zwei Durchgängen	139
11.5.5	Vermeidung von vertikalen Zusammenstößen	139
11.6	Horizontale Abstände	139
11.6.1	Überblick über horizontale Abstände	139
11.6.2	Eine neue Umgebung mit anderen Abständen	139

11.6.3	Horizontale Abstände verändern	139
11.6.4	Zeilenlänge	139
11.6.5	Proportionale Notation	139
12	Schnittstellen für Programmierer	140
12.1	Musikalische Funktionen	140
12.1.1	Überblick über musikalische Funktionen	140
12.1.2	Einfache Ersetzungsfunktionen	140
12.1.3	Paarige Ersetzungsfunktionen	140
12.1.4	Mathematik in Funktionen	140
12.1.5	Leere Funktionen	140
12.1.6	Funktionen ohne Argumente	140
12.1.7	Überblick über vorhandene musikalische Funktionen	140
12.2	Schnittstelle für Programmierer	143
12.2.1	Eingabevariablen und Scheme	143
12.2.2	Interne Repräsentation der Musik	143
12.3	Komplizierte Funktionen erstellen	143
12.3.1	Musikalische Funktionen darstellen	143
12.3.2	Eigenschaften von Musikobjekten	143
12.3.3	Verdopplung einer Note mit Bindebögen (Beispiel)	143
12.3.4	Artikulationszeichen zu Noten hinzufügen (Beispiel)	143
12.4	Programmierungsschnittstelle für Textbeschriftungen	143
12.4.1	Beschriftungskonstruktionen in Scheme	143
12.4.2	Wie Beschriftungen intern funktionieren	143
12.4.3	Neue Definitionen von Beschriftungsbefehlen	144
12.4.4	New markup list command definition	144
12.5	Kontexte für Programmierer	144
12.5.1	Kontextauswertung	144
12.5.2	Eine Funktion auf alle Layout-Objekte anwenden	144
12.6	Scheme-Vorgänge als Eigenschaften	144
Anhang A	Literatur	145
Anhang B	Scheme-Übung	146
Anhang C	Notationsübersicht	147
C.1	Liste der Akkordbezeichnungen	147
C.2	MIDI-Instrumente	147
C.3	Liste der Farben	147
C.4	Die Feta-Schriftart	147
C.5	Notenkopfstile	147

Anhang D	Vorlagen	148
D.1	Ein einzelnes System	148
D.1.1	Nur Noten	148
D.1.2	Noten und Text	148
D.1.3	Noten und Akkordbezeichnungen	148
D.1.4	Noten, Text und Akkordbezeichnungen	148
D.2	Klaviervorlagen	148
D.2.1	Piano Solo	148
D.2.2	Klavier und Gesangstimme	148
D.2.3	Klavier mit zentriertem Text	148
D.2.4	Klavier mit zentrierten Lautstärkebezeichnungen	148
D.3	Streichquartett	148
D.3.1	Streichquartett	148
D.3.2	Streichquartettstimmen	148
D.4	Vokalensemble	148
D.4.1	SATB Partitur	148
D.4.2	SATB Partitur und automatischer Klavierauszug	148
D.4.3	SATB mit zugehörigen Kontexten	148
D.5	Vorlagen für alte Notation	148
D.5.1	Transkription mensuraler Musik	148
D.5.2	Vorlage zur Transkription von Gregorianik	148
D.6	Jazz combo	148
D.7	Lilypond-book-Vorlagen	149
D.7.1	LaTeX	149
D.7.2	Texinfo	149
Anhang E	Befehlsübersicht	150
Anhang F	GNU Free Documentation License	154
F.0.1	Anhang: Wie kann die Lizenz für eigene Dokumente verwendet werden	159
Anhang G	Index der LilyPond-Befehle	160
Anhang H	LilyPond-Index	161

Vorwort

Es muss wohl während einer Probe des EJE (Eindhovens Jugendorchester) etwa 1995 gewesen sein, als Jan, einer der schrägen Bratschisten, Han-Wen, einem der verstimmten Hornisten, von seinem großartigen neuen Projekt erzählte, an dem er gerade arbeitete. Es sollte ein automatisiertes System für den Musikdruck werden (um genauer zu sein, es war MPP, ein Präprozessor für MusiXTeX). Wie der Zufall so will, hatte Han-Wen einige Stimmauszüge von einer Partitur, die er setzen wollte, und so schaute er sich das Programm an und war schnell begeistert. Man entschied sich, dass MPP in eine Sackgasse führte, und nach vielem Philosophieren und hitzigem E-Mail-Austausch begann Han-Wen mit LilyPond 1996. Dieses Mal wurde Jan mitgerissen von Han-Wens neuem Projekt.

Die Entwicklung eines Computerprogramms erinnert in vielem an das Erlernen eines Musikinstrumentes. Am Anfang macht es Spaß herauszufinden, wie alles funktioniert und alles, was man noch nicht kann, stellt eine Herausforderung dar. Nach der ersten Begeisterung muss man jedoch viel üben. Tonleitern und Etüden können furchtbar langweilig sein, und wenn keine Ermunterung von anderen – Lehrern, Dirigenten oder dem Publikum – kommt, ist es oft eine große Versuchung, einfach aufzuhören. Aber man macht weiter, und langsam wird das Instrument zu einem Teil des eigenen Lebens. An manchen Tagen geht alles wie von selbst und es macht Spaß, an anderen Tagen ist es nur Arbeit, aber man macht trotzdem weiter, jeden Tag.

Die Arbeit an LilyPond kann genauso wie das Spiel eines Instruments sehr langweilig sein, und manchmal kommt es vor lauter Fehlern so vor, als stapfe man durch einen Morast. Trotzdem ist die Arbeit schon Teil unseres Lebens geworden und wir machen einfach weiter. Die wahrscheinlich wichtigste Motivation ist wohl, dass unser Programm wirklich nützlich ist. Wenn wir im Internet surfen, finden wir viele Leute, die LilyPond benutzen und damit außerordentlich beeindruckende Partituren erstellen. Das zu sehen fühlt sich auf angenehme Weise etwas unwirklich an.

Unsere Stimmung heben aber nicht nur die Benutzer unseres Programmes, sondern auch die vielen Menschen, die uns helfen, indem sie Vorschläge einbringen, auf verschiedene Art an LilyPond mitwirken oder Fehlerberichte schicken. Ihnen allen möchten wir hier Dank sagen!

Musik spielen und Musik zu Papier zu bringen ist mehr als eine nette Analogie. Zusammen zu programmieren macht viel Spaß und Menschen helfen zu können ist sehr zufriedenstellend, aber letzten Endes geht es uns darum, durch dieses Programm unsere Liebe zur Musik auszudrücken. Wir hoffen, Sie können viele schöne Partituren damit setzen!

Han-Wen und Jan

Utrecht/Eindhoven, Die Niederlande, Juli 2002.

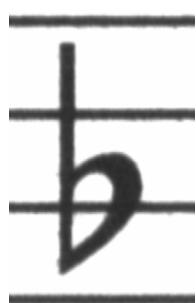
1 Einleitung

1.1 Notensatz

Die Kunst des Notensatzes wird auch als Notenstich bezeichnet. Dieser Begriff stammt aus dem traditionellen Notendruck. Noch etwa bis vor 20 Jahren wurden Noten erstellt, indem man sie in eine Zink- oder Zinnplatte schnitt oder mit Stempeln schlug. Diese Platte wurde dann mit Druckerschwärze versehen, so dass sie in geschnittenen und gestempelten Vertiefungen blieb. Diese Vertiefungen schwärzten dann ein auf die Platte gelegtes Papier. Das Gravieren wurde vollständig von Hand erledigt. Es war darum sehr mühsam, Korrekturen anzubringen, weshalb man von vornherein richtig schneiden musste. Es handelte sich dabei um ein sehr spezialisiertes Handwerk.

Heutzutage wird fast alle gedruckte Musik von Computern erstellt. Das hat einige deutliche Vorteile: Drucke sind billiger als die gravierten Platten und der Computersatz kann per E-Mail verschickt werden. Leider hat der intensive Einsatz des Computers die graphische Qualität des Notensatzes vermindert. Mit dem Computer erstellte Noten sehen langweilig und mechanisch aus, was es erschwert, von ihnen zu spielen.

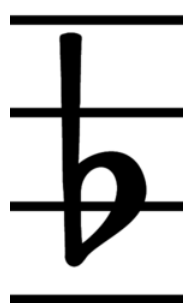
Die Abbildung unten illustriert den Unterschied zwischen traditionellem Notensatz und einem typischen Computersatz. Das dritte Bild zeigt, wie LilyPond die Formen des traditionellen Satzes nachahmt. Das linke Bild zeigt ein eingescanntes b-Vorzeichen aus einer 2000 herausgegebenen Edition. Das mittlere Bild zeigt das b-Vorzeichen der selben Musik aus einer handgestochenen Bärenreiter-Ausgabe. Das linke Bild zeigt die typischen Makel des Computer-Satzes: Die Notenlinien sind sehr dünn, die Schwärze des Vorzeichens entspricht den dünnen Linien und hat eine gerade Form mit scharfen Ecken und Kanten. Im Gegensatz dazu hat das Bärenreiter-Vorzeichen dicke, gerade zu sinnlich rundliche Formen. Unser Symbol für das Vorzeichen hat neben anderen auch dieses b als Vorbild. Es ist abgerundet und passt zu unseren Notenlinien, die sehr viel dicker sind als die der entsprechenden Computer-Ausgabe.



Henle (2000)



Bärenreiter (1950)

LilyPond
Feta-Schriftart
(2003)

Die Verteilung der Noten innerhalb des Taktes sollte ihrer Dauer entsprechen. Moderne Partituren zeigen diese Verhältnisse jedoch mit einer mathematischen Präzision, die nur sehr schlechte Ergebnisse bringt. Im nächsten Beispiel ist ein Motiv zweimal gesetzt: einmal mit den exakten mathematischen Längenverhältnissen, dann mit kleinen Korrekturen. Welches von beiden ist mit dieser Korrektur gesetzt?





In diesem Ausschnitt kommen nur Viertel vor, Noten, die in einem gleichmäßigen Rhythmus gespielt werden. Die Abstände sollten das widerspiegeln. Leider lässt uns aber das Auge im Stich: es beachtet nicht nur den Abstand von aufeinander folgenden Notenköpfen, sondern auch den ihrer Hälse. Also müssen Noten, deren Hälse in direkter Folge zuerst nach oben und dann nach unten ausgerichtet sind, weiter auseinander gezogen werden, während die unten/oben-Folge engere Abstände fordert, und das alles auch noch in Abhängigkeit von der vertikalen Position der Noten. Das obere Beispiel ist mit dieser Korrektur gesetzt, das unter ohne. In letzterem Fall bilden sich für das Auge bei unten/oben-Folgen Notenklumpen mit schmalen Abständen zwischen den Notenhälsen.

Musiker sind üblicherweise zu sehr konzentriert, die Musik aufzuführen, als dass sie das Aussehen der Noten studieren könnten; und diese Erbsenzählerei der typographischen Details mag akademisch wirken. Das ist aber nicht gerechtfertigt. Unser Beispielstück hat einen monotonen Rhythmus, und wenn alle Zeilen gleich aussehen, wird das Notenblatt zu einem Labyrinth. Wenn der Spieler auch nur einmal wegschaut oder kurze Zeit unkonzentriert ist, findet er nicht mehr zurück zu der Stelle, an der er war.

Der dichtere Eindruck, den die dickeren Notenlinien und schwereren Notationssymbole schaffen, eignet sich auch besser für Noten, die weit vom Leser entfernt stehen, etwa auf einem Notenständer. Eine sorgfältige Verteilung der Zwischenräume erlaubt es, die Noten sehr dicht zu setzen, ohne dass die Symbole zusammenklumpen. Dadurch werden unnötige Seitenumbrüche vermieden, sodass man nicht so oft blättern muss.

Hier eine übliche Charakteristik der Typographie: Das Layout sollte schön sein, nicht um seiner selbst willen, sondern um dem Leser zu helfen. Für Aufführungsmaterial wie Partituren ist das um so wichtiger. Ein Spieler kann den Noten nur eine begrenzte Aufmerksamkeit schenken. Und je weniger Aufmerksamkeit nötig ist, um die Noten zu erfassen, um so mehr Zeit können sie in die Aufführung selber stecken. So wirkt sich gute Typographie direkt in eine verbesserte Aufführung aus!

Die Beispiele haben gezeigt, dass der Notensatz eine subtile und komplexe Kunst ist und gute Ergebnisse nur mit viel Erfahrung erlangt werden können, die Musiker normalerweise nicht haben. LilyPond stellt unser Bemühen dar, die graphische Qualität handgestochener Notenseiten ins Computer-Zeitalter zu transportieren und sie für normale Musiker erreichbar zu machen. Wir haben unsere Algorithmen, die Gestalt der Symbole und die Programm-Einstellungen darauf abgestimmt, einen Ausdruck zu erzielen, der der Qualität der alten Editionen entspricht, die wir so gerne betrachten und von denen wir gerne spielen.

1.2 Automatisierter Notensatz

Wie sollen wir also jetzt die Typographie anwenden? Wie können wir erwarten, dass wir in der Lage wären, ein Programm zu schreiben, dass den Beruf des Notenstechers ersetzt, wo dieser doch mehr als zehn Jahre braucht, um ein Meister zu werden?

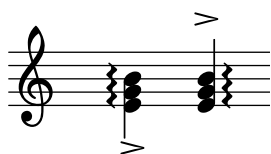
Wir können es tatsächlich nicht! Da Typographie allein durch das menschliche Auge bestimmt ist, kann der Mensch nicht ersetzt werden. Aber sehr viel mechanische Arbeit kann erleichtert werden. Indem etwa LilyPond die üblichen Situationen kennt und bewältigt, können die restlichen Fehler von Hand beseitigt werden. Das ist schon ein großer Fortschritt im Vergleich mit den existierenden Programmen. Und mit der Zeit können immer mehr Fälle automatisiert werden, so dass immer weniger Eingriffe von Hand notwendig werden.

Als wir anfangen, haben wir ein Programm mit C++ geschrieben. Das heißt aber, dass der Funktionsumfang des Programms vom Anfang an durch die Entwickler festgelegt ist. Das erschien uns aber nicht ausreichend:

- Wenn LilyPond Fehler macht, muss der Benutzer die Einstellungen ändern können. Er muss also Zugang zur Formatierungsmaschinerie haben. Also können die Regeln und Einstellungen nicht beim Kompilieren des Programms festgelegt werden, sondern sie müssen während des Laufes zugänglich sein.
- Notensatz ist eine Frage des Augenmaßes, und damit auch vom Geschmack abhängig. Benutzer können mit unseren Entscheidungen unzufrieden sein. Darum müssen also auch die Definitionen des typographischen Stils dem Benutzer zugänglich sein.
- Schließlich verfeinern wir unseren Formatierungsalgorithmus immer weiter, also müssen die Regeln auch flexibel sein. Die C++-Sprache zwingt zu einer bestimmten Gruppierungsmethode, die nicht den Regeln für den Notensatz entspricht.

Diese Probleme wurden angegangen, indem ein Übersetzer für die Scheme-Programmiersprache integriert wurde und Teile von LilyPond in Scheme neu geschrieben wurden. Die derzeitige Formatierungsarchitektur ist um die Notation von graphischen Objekten herum aufgebaut, die von Scheme-Variablen und -Funktionen beschrieben werden. Diese Architektur umfasst Formatierungsregeln, typographische Stile und individuelle Formatierungsentscheidungen. Der Benutzer hat direkten Zugang zu den meisten dieser Einstellungen.

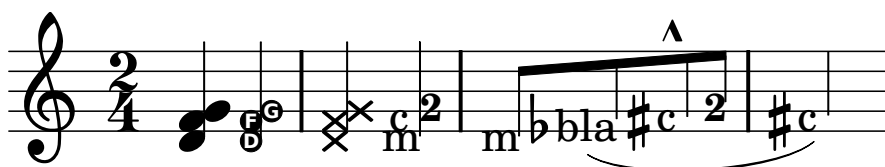
Scheme-Variablen steuern Layout-Entscheidungen. Zum Beispiel haben viele graphische Objekte eine Richtungsvariable, die zwischen oben und unten (oder rechts und links) wählen kann. Hier etwa sind zwei Akkorde mit Akzenten und Arpeggien. Beim ersten Akkord sind alle Objekte nach unten (oder links) ausgerichtet, beim zweiten nach oben (rechts).



Der Prozess des Notensatzes besteht für das Programm darin, die Variablen der graphischen Objekte zu lesen und zu schreiben. Einige Variablen haben festgelegte Werte. So ist etwa die Dicke von vielen Linien – ein Charakteristikum des typographischen Stils – von vornherein festgelegt. Wenn sie geändert werden, ergibt sich ein anderer typographischer Eindruck.



Formatierungsregeln sind auch von vornherein festgelegt. Jedes Objekt hat Variablen, die Vorgänge beschreiben. Diese Vorgänge machen die eigentlich Satzarbeit, und wenn man sie durch andere ersetzt, kann ihr Verhalten verändert werden. Im nächsten Beispiel wird die Regel, mit der die Notenköpfe gezeichnet werden, verändert.



1.3 Welche Symbole?

Während des Notensatzprozesses entscheidet sich, wo Symbole platziert werden. Das kann aber nur gelingen, wenn vorher entschieden wird, *welche* Symbole gesetzt werden sollen, also welche Notation benutzt werden soll.

Die heutige Notation ist ein System zur Musikaufzeichnung, das sich über die letzten 1000 Jahre entwickelt hat. Die Form, die heute üblicherweise benutzt wird, stammt aus dem frühen Barock. Auch wenn sich die grundlegenden Formen (also die Notenköpfe, das Fünfliniensystem) nicht verändert hat, entwickeln sich die Details trotzdem immer noch weiter, um die Errungenschaften der Neuen Musik darstellen zu können. Die Notation umfasst also 500 Jahre Musikgeschichte. Ihre Anwendung reicht von monophonen Melodien bis zu ungeheurem Kontrapunkt für großes Orchester.

Orchester. Wie bekommen wir dieses vielköpfige Monster zu fassen? Unsere Lösung ist es, eine strikte Trennung zwischen der Notation, also welche Symbole benutzt werden, und dem Satz, also wohin sie gesetzt werden, zu machen. Um das Problem anzupacken, haben wir es in kleine (programmierbare) Happen zerteilt, so dass jede Art von Symbol durch ein eigenes Plugin verarbeitet wird. Alle Plugins kooperieren durch die LilyPond-Architektur. Sie sind vollständig modular und unabhängig und können somit auch unabhängig voneinander entwickelt werden. Der Schreiber, der die Musik in Graphik umwandelt, ist ein Kopist oder Notenstecher (engl. engraver). Darum werden die Plugins als **engraver** bezeichnet.

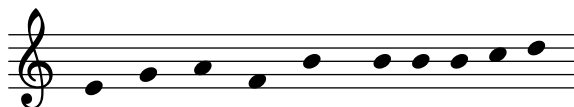
Im nächsten Beispiel wird gezeigt, wie mit dem Plugin für die Notenköpfe, dem `Note_heads_engraver` („Notenkopfstecher“) der Satz begonnen wird.



Dann fügt ein `Staff_symbol_engraver` („Notensystemstecher“) die Notenlinien hinzu.



Der `Clef_engraver` („Notenschlüsselstecher“) definiert eine Referenzstelle für das System.



Der `Stem_engraver` („Halsstecher“) schließlich fügt Hälse hinzu.



Dem `Stem_engraver` wird jeder Notenkopf mitgeteilt, der vorkommt. Jedes Mal wenn ein (oder mehrere bei einem Akkord) Notenkopf erscheint, wird ein Hals-Objekt erstellt und an den Kopf geheftet. Wenn wir dann noch engraver für Balken, Bögen, Akzente, Vorzeichen, Taktlinien, Taktangaben und Tonartbezeichnungen hinzufügen, erhalten wir eine vollständige Notation.



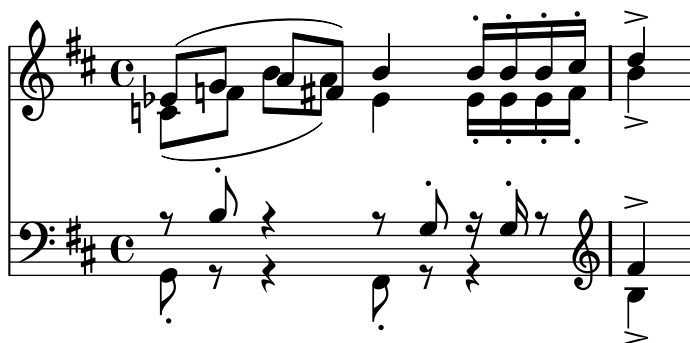
Dieses System funktioniert gut für monophone Musik, aber wie geht es mit Polyphonie? Hier müssen sich mehrere Stimmen ein System teilen.



In diesem Fall werden das System und die Vorzeichen geteilt, aber die Hälse, Bögen, Balken usw. sind jeder einzelnen Stimme eigen. Die engraver müssen also gruppiert werden. Die Köpfe, Hälse, Bögen usw. werden in einer Gruppe mit dem Namen „Voice context“ (Stimmenkontext) zusammengefasst, die engraver für den Schlüssel, die Vorzeichen, Taktstriche usw. dagegen in einer Gruppe mit dem Namen „Staff context“ (Systemkontext). Im Falle von Polyphonie hat ein Staff context dann also mehr als einen Voice context. Auf gleiche Weise können auch mehrere Staff contexts in einen großen Score context (Partiturkontext) eingebunden werden.

Siehe auch

Programmreferenz: `Contexts`.



1.4 Musikalische Repräsentation

Idealerweise ist das Eingabeformat für ein höheres Satzsystem die abstrakte Beschreibung des Inhaltes. In diesem Fall wäre das die Musik selber. Das stellt uns aber vor ein ziemlich großes Problem, denn wie können wir definieren, was Musik wirklich ist? Anstatt darauf eine Antwort zu suchen, haben wir die Frage einfach umgedreht. Wir schreiben ein Programm, das den Notensatz beherrscht und passen das Format an, so einfach wie möglich zu sein. Wenn es nicht mehr vereinfacht werden kann, haben wir per Definition nur noch den reinen Inhalt. Unser Format dient als die formale Definition eines Musiktextes.

Die Syntax ist gleichzeitig die Benutzerschnittstelle bei LilyPond, darum soll sie einfach zu schreiben sein; z. B. bedeutet

`c'4 d'8`

eine Viertel `c'` und eine Achtel `d'`, wie in diesem Beispiel:



In kleinem Rahmen ist diese Syntax sehr einfach zu benutzen. In größeren Zusammenhängen aber brauchen wir Struktur. Wie sonst kann man große Opern oder Symphonien notieren? Diese Struktur wird gewährleistet durch sog. music expressions (Musikausdrücke): indem kleine Fragmente zu größeren kombiniert werden, kann mehr Komplexität ausgedrückt werden. So etwa hier:

```
c4
```



Gleichzeitig erklingende Noten werden hinzugefügt, indem man alle in << und >> einschließt.

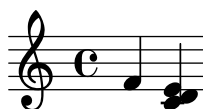
```
<<c4 d4 e4>>
```



Um aufeinanderfolgende Noten darzustellen, werden sie in geschweifte Klammern gefasst:

```
{ ... }
```

```
{ f4 <<c4 d4 e4>> }
```



Dieses Gebilde ist in sich wieder ein Ausdruck, und kann daher mit einem anderen Ausdruck kombiniert werden (hier mit einer Halben).

```
<< g2 \ { f4 <<c4 d4 e4>> } >>
```



Solche geschachtelten Strukturen können sehr gut in einer kontextunabhängigen Grammatik beschrieben werden. Der Programmcode für den Satz ist auch mit solch einer Grammatik erstellt. Die Syntax von LilyPond ist also klar und ohne Zweideutigkeiten definiert.

Die Benutzerschnittstelle und die Syntax werden als erstes vom Benutzer wahrgenommen. Teilweise ist es eine Frage des Geschmacks und auch ein Objekt vieler Diskussionen. Auch wenn Geschmacksfragen ihre Berechtigung haben, sind sie nicht sehr produktiv. Im großen Rahmen von LilyPond spielt die Eingabe-Syntax nur eine geringe Rolle, denn eine logische Syntax zu schreiben ist einfach, guten Formatierungscode aber sehr viel schwieriger. Das kann auch die Zeilenzahl der Programmzeilen zeigen: Analysieren und Darstellen nimmt nur etwa 10% des Codes ein:

1.5 Beispielanwendung

Wir haben LilyPond als einen Versuch geschrieben, wie man die Kunst des Musiksatzes in ein Computerprogramm komprimieren kann. Dieses Programm kann nun dank vieler harter Arbeitsstunden benutzt werden, um sinnvolle Aufgaben zu erledigen. Die einfachste ist dabei der Notendruck.



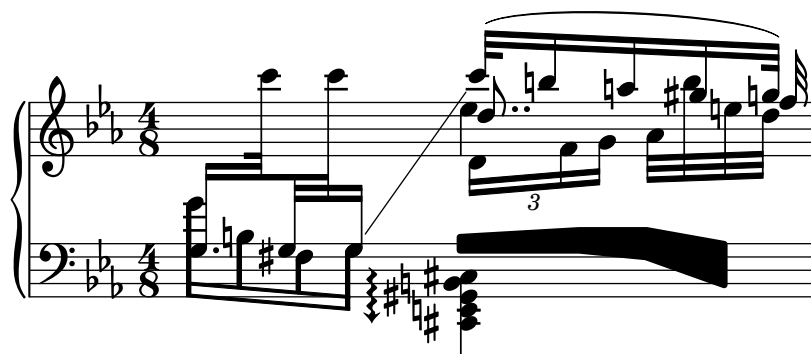
Indem wir Akkordsymbole und einen Text hinzufügen, erhalten wir ein Lead Sheet.



Mehrstimmige Notation und Klaviermusik kann auch gesetzt werden. Das nächste Beispiel zeigt einige etwas exotischere Konstruktionen:

Screech and boink Random complex notation

Han-Wen Nienhuys



Dieses Beispiel wurde vollständig selbst geschrieben, aber das ist nicht nötig. Da der Satz fast vollständig automatisch abläuft, kann er auch als eine Ausgabe-Erweiterung für andere Programme dienen, die Musik manipulieren. So können etwa ganze Datenbanken musikalischer Fragmente automatisch in Notenbilder umgewandelt werden, die dann auf Internetseiten oder in Multimediapräsentation Anwendung finden.

Dieses Benutzerhandbuch zeigt eine weitere Möglichkeit: Das Eingabeformat der Noten ist reiner Text, darum könne sie sehr einfach integriert werden in andere textbasierte Formate wie etwa \LaTeX , HTML oder, wie in diesem Fall, Texinfo. Durch ein spezielles Programm werden die Eingabefragmente durch Notenbilder in der resultierenden PDF- oder HTML-Datei ersetzt. Dadurch ist es sehr einfach, Musik und Text zu kombinieren.

1.6 Über das Handbuch

Das Handbuch ist in folgende Kapitel gegliedert:

- *Kapitel 2 [Übung], Seite 11* stellt eine einfache Einführung in den Musiksatz dar. Neulinge sollten hiermit beginnen.
- *Kapitel 3 [Alles zusammenfügen], Seite 35* erklärt generelle Konzepte des LilyPond-Dateiformates. Wenn Sie sich nicht sicher sind, wohin ein Befehl gesetzt werden soll, lesen Sie hier nach.
- *Kapitel 4 [An LilyPond-Projekten arbeiten], Seite 44* zeigt den wirklichen Einsatz von LilyPond und gibt Hinweise, wie einige Probleme vermieden werden können.
- *Kapitel 5 [Die Ausgabe verändern], Seite 53* stellt dar, wie die Standardeinstellungen von LilyPond verändert werden können.
- *Kapitel 6 [Grundlegende Notation], Seite 62* erklärt alles über die grundlegenden Notationskonstruktionen. Dieses Kapitel ist für fast jedes Notationsprojekt nützlich.
- *Kapitel 7 [Instrumentenspezifische Notation], Seite 117* erklärt spezifische Schwierigkeiten, die sich bei bestimmten Notationstypen ergeben. Dieses Kapitel ist nur in entsprechenden Fällen bestimmter Instrumente oder bei Gesang zu konsultieren.
- *Kapitel 8 [Fortgeschrittene Notationstechniken], Seite 122* erklärt komplizierte oder unübliche Anwendungen nach Notationsgegenstand geordnet.
- *Kapitel 9 [Standardeinstellungen verändern], Seite 134* erklärt, wie das Layout getrimmt werden kann.
- *Kapitel 10 [Nichtmusikalische Notation], Seite 136* zeigt alles, was nicht direkt mit den Noten zu tun hat wie Titel, mehrere Sätze oder wie man ein MIDI-Instrument auswählt.
- *Kapitel 11 [Abstände], Seite 138* befasst sich mit globalen Fragen wie der Definition von Papierformaten oder wie man Seitenumbrüche definiert.
- *Kapitel 12 [Schnittstellen für Programmierer], Seite 140* demonstriert die Erstellung von musikalischen Funktionen.
- *program usage manual, Running LilyPond* zeigt, wie LilyPond und die Hilfsprogramme gestartet werden. Zusätzlich wird hier gezeigt, wie Quelldateien von älteren LilyPond-Versionen aktualisiert werden können.
- *program usage manual, LilyPond-book* zeigt die Details der Integration von Noten in Texte wie etwa dieses Handbuch.
- *program usage manual, Converting from other formats* erklärt die Funktionsweise der Konvertierungsprogramme. Diese Programme sind im LilyPond-Paket enthalten und können ein ganze Anzahl von Formaten in das .ly-Format umwandeln.
- *Anhang A [Literatur], Seite 145* enthält einige wichtige Quellen für alle, die mehr über Notation und den Notensatz erfahren wollen.
- Das *Anhang B [Scheme-Übung], Seite 146* stellt eine kurze Einleitung in die Scheme-Sprache dar, mit dem die musikalischen Funktionen gebildet werden.
- *Anhang C [Notationsübersicht], Seite 147* sind Tabellen, in denen Akkordbezeichnungen, MIDI-Instrumente, Farbbezeichnungen und die Zeichen der Feta-Schriftart gesammelt sind.
- *Anhang D [Vorlagen], Seite 148* von LilyPond-Stücken. Kopieren Sie einfach hier, fügen Sie in ihre Datei ein und schreiben Sie noch die Noten dazu. Das ist alles!
- Die *Anhang E [Befehlsübersicht], Seite 150* zeigt die wichtigsten LilyPond-Befehle.
- Der *Anhang G [Index der LilyPond-Befehle], Seite 160* listet alle Befehle auf, die mit \ anfangen.
- Der *Anhang H [LilyPond-Index], Seite 161* ist ein vollständiger Index.

Wenn Sie etwas Erfahrung gesammelt haben, können Sie das Handbuch als Referenz benutzen, denn es hat einen sehr guten Index¹, aber es ist auch als eine große HTML-Seite, erhältlich, die einfach mit der Suchfunktion des Browsers durchsucht werden kann.

Wenn Sie auf musikalische Begriffen stoßen, die Ihnen nicht geläufig sind, kann vielleicht das Glossar helfen.

Hier werden die wichtigsten Begriffe auf englisch erklärt und in eine Reihe von Sprachen übersetzt, sodass sie auch auf deutsch gefunden werden. Es ist ein eigenes Dokument, das als HTML- oder PDF-Datei geladen werden kann.

Dieses Handbuch ist ohne eine Reihe anderer Dokumente nicht vollständig. Sie sind nicht in gedruckter Form erhältlich, aber sollten in dem Dokumentationspaket für Ihr Betriebssystem enthalten sein:

- Programmreferenz

Die Programmreferenz ist eine Sammlung intensiv verlinkter HTML-Seiten, die alle Details jeder einzelnen LilyPond-Klasse, jedes Objektes und jeder Funktion erklären. Sie wird direkt aus den Satzdefinitionen produziert.

So gut wie alle Formatierungsmöglichkeiten, die intern verwendet werden, sind auch direkt für den Benutzer zugänglich. Alle Variablen z. B., die Dicke-Werte, Entfernungen usw. kontrollieren, können in den Eingabe-Dateien verändert werden. Es gibt eine riesige Anzahl von Formatierungsoptionen, und alle haben einen ‚See also‘-Abschnitt, der auf die Dokumentation verweist. Im HTML-Handbuch haben diese Abschnitte klickbare Links.

Die Programmreferenz ist nur auf englisch erhältlich.

- Verschiedene Quelltextbeispiele.

Diese Dateisammlung zeigt einige Tricks und ist als eine große HTML-Datei herunterladbar. Notenbilder und Erklärungen sind enthalten.

- Die Regressionstests.

Diese Dateisammlung testet jede Notations- und Satzeigenschaft von LilyPond in einer besonderen Datei. Die Sammlung ist vor allem da, um die Fehlersuche zu vereinfachen, aber sie kann sehr informativ sein, um zu sehen, wie wir das Programm entwickeln. Das Format entspricht dem der Tipps-Seite.

Auf allen HTML-Seiten, die Noten eingebettet haben, deren Ausgabe mit LilyPond vorgenommen wurde, kann die originale Quelldatei durch einen Klick auf das Bild betrachtet werden.

Der Speicherort der Dokumentationsdateien unterscheidet sich evtl. je nach Betriebssystem. Manchmal wird hier auf Initialisierungs- oder Beispieldateien verwiesen. Das Handbuch nimmt dabei an, dass diese Dateien sich relativ zum Quellverzeichnis befinden. Zum Beispiel würde der Pfad `‘input/test/bla.ly’` etwa auf die Datei `‘lilypond2.x.y/input/test/bla.ly’` verweisen. In den Binärpaketen für Unix-Plattformen sind Dokumentation und Beispiele üblicherweise in einem Verzeichnis wie `‘/usr/share/doc/lilypond/’` gespeichert. Initialisierungsdateien, etwa `‘scm/lily.scm’`, oder `‘ly/engraver-init.ly’`, befinden sich normalerweise im Verzeichnis `‘/usr/share/lilypond/’`.

Dieses Handbuch (so wie alle anderen) ist als PDF- oder HTML-Datei von LilyPonds Internetseite <http://www.lilypond.org/> herunterladbar.

¹ Wenn Sie nach etwas suchen und es nicht im Handbuch finden, gilt das als Fehler des Handbuches. In diesem Fall geben Sie bitte einen Hinweis!

2 Übung

Diese Übung führt ein in die Notationssprache des Programmes LilyPond und erklärt, wie man damit Noten setzen kann. Nach einer ersten Einleitung wird erklärt, wie die häufigsten Notenbilder im Druck umgesetzt werden können.

2.1 Erste Schritte

In diesem Abschnitt werden die Grundlagen zur Benutzung des Programmes erklärt.

2.1.1 Eine Quelldatei übersetzen

Das erste Beispiel zeigt, wie LilyPond arbeitet. Um Noten zu erzeugen, muss man eine Textdatei schreiben, in der die Notation beschrieben wird. Zum Beispiel ergibt die Datei

```
{
  c' e' g' e'
}
```

folgendes Resultat.



Warnung: In jeder LilyPond-Datei müssen **{ geschweifte Klammern }** um die Noten gesetzt werden. Vor und hinter den Klammern sollten Leerzeichen eingegeben werden, damit keine Unklarheiten in Verbindung mit den eigentlichen Notensymbolen entstehen. An Anfang und Ende der Zeile können diese Leerzeichen auch weggelassen werden. Es kann sein, dass in diesem Handbuch die Klammern in manchen Beispielen fehlen, aber man sollte immer daran denken, sie in den eigenen Dateien zu benutzen!

Zusätzlich unterscheidet LilyPond **Groß- und Kleinschreibung**. `{ c d e }` ist zulässiger Code, `{ C D E }` dagegen resultiert in einer Fehlermeldung.

Eingabe von Noten und Ansicht des Ergebnisses

In diesem Kapitel zeigen wir, welche Kommandos eingegeben werden müssen und wie das Resultat dann betrachtet werden kann.

MacOS X

Wenn Sie das `LilyPond.app`-Symbol doppelt klicken, öffnet sich eine Beispiel-Datei. Speichern Sie sie etwa als `test.ly` auf dem Desktop und übersetzen Sie sie mit dem Menü-Befehl „Compile > Typeset File“. Die PDF-Datei mit dem fertigen Notensatz wird automatisch geöffnet.

Das erste Mal, wenn LilyPond aufgerufen wird, kann es etwa eine Minute dauern, bis die Eingabedatei übersetzt wird, weil die Schriften des Betriebssystems erst überprüft werden müssen.

Das nächste Mal, wenn Sie LilyPond benutzen, sollten Sie „New“ oder „Open“ wählen. Sie müssen die Datei speichern, bevor Sie sie übersetzen können. Wenn es Fehler gibt, lesen Sie die Meldungen im Log-Fenster.

Windows

Wenn sie auf das LilyPond-Symbol auf dem Desktop doppelklicken, öffnet sich ein einfacher Texteditor mit einer Beispieldatei. Speichern Sie sie z. B. als `test.ly` auf dem Desktop und klicken Sie dann doppelt auf die Datei, um die Übersetzung zu beginnen. Nach einigen Sekunden wird eine Datei `test.pdf` auf dem Desktop erscheinen. Mit einem Doppelklick kann das fertige Notenbild in der PDF-Datei angezeigt werden. Eine Alternative ist es, die `test.ly`-Datei mit der Maus auf das LilyPond-Symbol zu ziehen.

Um eine schon existierende Datei zu bearbeiten, klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie „Edit source“. Um eine leere Datei zu erhalten, mit der Sie ein neues Stück beginnen können, öffnen Sie den Texteditor durch Doppelklick auf das LilyPond-Symbol und benutzen Sie „New“ im „File“-Menü.

Mit dem Doppelklick wird nicht nur die PDF-Datei erstellt, sondern auch eine `.log`-Datei. Hier wird gespeichert, was LilyPond aus der Quelldatei gelesen hat. Sollten Fehler auftreten, hilft oft ein Blick in diese Datei.

Es gibt einige andere Editoren mit besserer Unterstützung für LilyPond; Einzelheiten dazu können Sie im Kapitel `program usage manual`, `Editor support` nachlesen.

Unix

Öffnen Sie ein Kommandozeilenfenster und einen Texteditor. Zum Beispiel können Sie `xterm` öffnen und `joe` ausführen.¹ Die einfachste Bearbeitungsgebung ist das `LilyPondTool`. Siehe auch `program usage manual`, `Editor support` für mehr Information. Geben Sie folgendes in Ihrem Editor ein und speichern Sie die Datei als `test.ly`:

```
{
  c' e' g' e'
}
```

Um die Datei zu bearbeiten, geben sie an der Konsole

```
lilypond test.ly
```

ein. Sie werden ungefähr folgende Meldungen sehen:

```
user@domain:~$ lilypond test.ly
GNU LilyPond 2.11.23
»test.ly« wird verarbeitet
Analysieren...
Interpretation der Musik...
Vorverarbeitung der grafischen Elemente...
Layout nach »test.ps« ausgeben...
Konvertierung nach »test.pdf«...
```

Als Ergebnis erhalten Sie ein `test.pdf`, das Sie mit den Standardprogrammen Ihres Betriebssystems anschauen können.²

2.1.2 Einfache Notation

LilyPond fügt einige Bestandteile des Notenbildes automatisch hinzu. Im nächsten Beispiel sind nur drei Tonhöhen angegeben, aber LilyPond setzt trotzdem einen Schlüssel, eine Taktangabe und Notendauern.

```
{
  c' e' g' e'
```

¹ Es gibt Makros für VIM-Süchtige und es gibt einen `LilyPond-Modus` für Emacs. Wenn diese noch nicht installiert sind, lesen Sie die Datei `INSTALL.txt`.

² Wenn derartige Programme nicht installiert sind, können Sie `Ghostscript` ausprobieren, ein frei erhältliches Paket zur Ansicht von PDF- und PostScript-Dateien.

}



Diese Einstellungen können verändert werden, aber in den meisten Fällen sind die automatischen Werte durchaus brauchbar.

Tonhöhen

Die Tonhöhen werden mit Kleinbuchstaben eingegeben, die den Notennamen entsprechen. Es ist jedoch wichtig zu wissen, dass LilyPond in seiner Standardeinstellung die englischen Notennamen verwendet. Bis auf eine Ausnahme entsprechen sie den deutschen, deshalb wird die Voreinstellung von LilyPond für diese Übung beibehalten. Die *Ausnahme* ist das h – in LilyPond muss man anstelle dessen b schreiben! Das deutsche b dagegen wird als bes notiert, ein his dagegen würde bis geschrieben. Siehe auch Kap. [Abschnitt 6.1.2 \[Versetzungszeichen\]](#), Seite 63 und [Abschnitt 6.1.5 \[Notenbezeichnungen in anderen Sprachen\]](#), Seite 65, hier wird beschrieben, wie sich die deutschen Notennamen benutzen lassen.

Am einfachsten können Noten im `\relative`-Modus eingegeben werden. In diesem Modus wird automatisch angenommen, dass das Intervalle zwischen der vorhergehenden und der aktuellen Note höchstens eine *Quarte* beträgt. Fangen wir unser erstes Notationsbeispiel mit einer Tonleiter an.

```
\relative c' {
  c d e f
  g a b c
}
```



Die erste Note ist ein eingestrichenes C. Jede folgende Note befindet sich höchstens eine Quarte von der vorhergehenden entfernt – das erste ,C' ist also das nächste C vom eingestrichenen C aus gerechnet. Darauf folgt das nächstmögliche D in Bezug auf die vorhergehende Note. Mit diesen Regeln können auch Melodien mit größeren Intervallen gebildet werden:

```
\relative c' {
  d f a g
  c b f d
}
```



Dieses Beispiel beginnt nicht mit dem eingestrichenen C. Die erste Note (das ,D') ist das nächste D vom eingestrichenen C aus gerechnet.

Um Intervalle zu notieren, die größer als eine Quarte sind, können wir die Oktave verändern. Mit einem Apostroph ' (Taste Shift+#) direkt hinter dem Notennamen wird die Oktave um eins erhöht, mit einem Komma , um eins erniedrigt.

```
\relative c' {
  a a, c' f,
  g g' a,, f'
}
```



Um eine Notenhöhe um zwei oder mehr Oktaven zu verändern, werden sukzessive ' oder ,, benutzt – es muss sich dabei wirklich um zwei einzelne Apostrophen und nicht um das Anführungszeichen " (Taste Shift+2) handeln. Auch die Anfangsoktave für einen \relative c'-Abschnitt kann so verändert werden.

Tondauern (Rhythmen)

Die **Tondauer** einer Note wird durch eine Zahl bezeichnet, die direkt auf den Notennamen folgend eingegeben wird. ,1‘ für eine **ganze Note**, ,2‘ für eine **halbe Note**, ,4‘ für eine **Viertelnote** und so weiter. Notenhäse werden automatisch hinzugefügt.

```
\relative c' {
  a1
  a2 a4 a8 a
  a16 a a a a32 a a a a64 a a a a a a a a2
}
```



Wenn keine Dauer bezeichnet wird, wird die der vorhergehenden Note verwendet. Für die erste Note ist eine Viertel als Standard definiert.

Um **punktierte Noten** zu erzeugen, wird einfach ein Punkt ‚.‘ hinter die Notendauer geschrieben.

```
\relative c'' {
  a a a4. a8
  a8. a16 a a8. a8 a4.
}
```



Pausen

Eine Pause wird genauso wie eine Note eingegeben; ihre Bezeichnung ist ,r'.

```
\relative c'' {
  a r r2
  r8 a r4 r4. r8
}
```



Taktangabe

Die Taktangabe kann mit dem \time-Befehl definiert werden:

```
\relative c'' {
  \time 3/4
  a4 a a
  \time 6/8
  a4. a
  \time 4/4
  a4 a a a
}
```



Notenschlüssel

Der Notenschlüssel kann mit dem \clef-Befehl gesetzt werden:

```
\relative c' {
  \clef treble
  c1
  \clef alto
  c1
  \clef tenor
  c1
  \clef bass
  c1
}
```



Alles zusammen

Hier ist ein kleines Beispiel, dass all diese Definitionen beinhaltet:

```
\relative c, {
  \time 3/4
  \clef bass
```

```

c2 e8 c' g'2.
f4 e d c4 c, r4
}

```



Mehr Information

Eingabe von Tonhöhen und -dauern

siehe [Abschnitt 6.1 \[Tonhöhen\]](#), Seite 62 und [Abschnitt 6.2.1 \[Tondauern\]](#), Seite 70.

Pausen siehe [Abschnitt 6.1.9 \[Pausen\]](#), Seite 68.

Taktangeben und andere Zeitangaben

siehe [Abschnitt 6.4.3 \[Taktangabe\]](#), Seite 84.

Notenschlüssel

siehe [Abschnitt 6.4.1 \[Notenschlüssel\]](#), Seite 81.

2.1.3 Arbeiten an Text-Dateien

LilyPonds Quelldateien werden wie Dateien in den meisten Programmiersprachen behandelt: Es ist auf Groß- und Kleinschreibung zu achten, Ausdrücke werden mit geschweiften Klammern { } eingeklammert und Kommentare mit dem Prozentzeichen % auskommentiert oder mit %{ ... }% umgeben.

Wenn das jetzt unverständlich erscheint, sind hier die Erklärungen:

- **Groß- und Kleinschreibung:** Die Bedeutung eines Zeichens verändert sich, je nachdem, ob es groß (A, B, S, T) oder klein (a, b, s, t) geschrieben wird. Noten müssen immer klein geschrieben werden, { c d e } funktioniert, während { C D E } einen Fehler produziert.
- **Leerzeichen:** Es spielt keine Rolle, wie viele Leerzeichen oder leere Zeilen sich zwischen den Zeichen der Quelldatei befinden. { c d e } bedeutet das Gleiche wie { c d e } oder

```

{
c                               d
e }

```

Natürlich ist das letzte Beispiel etwas schwer zu lesen. Eine gute Daumenregel ist es, Code-Blöcke mit der Tab-Taste oder zwei Leerzeichen einzurücken,

```

{
  c d e
}.

```

- **Ausdrücke:** Auch der kleinste Abschnitt an LilyPond-Code muss in { **geschweifte Klammern** } eingeschlossen werden. Diese Klammern zeigen LilyPond an, dass es sich um einen zusammengehörenden musikalischen Ausdruck handelt, genauso wie Klammern ,()' in der Mathematik. Die Klammern sollten von jeweils einem Leerzeichen umgeben sein, um Zweideutigkeiten auszuschließen, es sei denn, sie befinden sich am Anfang oder Ende einer Zeile. Eine Funktion (wie etwa `\relative { }`) wird auch als ein einzelner Musikausdruck gewertet.
- **Kommentare:** Ein Kommentar ist eine Bemerkung für den menschlichen Leser einer Quelldatei, es wird bei der Dateianalyse durch das Programm ignoriert, so dass es also keine Auswirkung auf die Druckausgabe der Noten hat. Es gibt zwei verschiedene Typen von Kommentaren. Das Prozentzeichen ,%' geht einem Zeilen-Kommentar voraus: Alles nach

diesem Zeichen wird in dieser Zeile ignoriert. Ein Block-Kommentar ist ein ganzer Abschnitt mit einem Kommentar. Alles, was von `%{` und `%}` umgeben ist, wird ignoriert. Das folgende Beispiel zeigt mögliche Anwendung von Kommentaren:

```
% Noten für twinkle twinkle hier
c4 c g' g a a g2

%{
  Diese Zeilen, und die Noten unten werden
  ignoriert, weil sie sich in einem Block-Kommentar
  befinden.

  g g f f e e d d c2
%}
```

Weitere Vorschläge zur Konstruktion von Quelldateien finden sich in [Abschnitt 4.1 \[Vorschläge\]](#), Seite 44.

2.1.4 Wie soll man die Übung lesen

Wie wir in [Abschnitt 2.1.3 \[Arbeiten an Text-Dateien\]](#), Seite 16 gesehen haben, muss LilyPond-Code immer von `{ }` Zeichen oder einem `\relative c' ' { ... }` umgeben sein. Im Rest dieses Handbuchs werden die meisten Beispiele allerdings darauf verzichten.

Wenn Sie in der HTML-Version der Dokumentation lesen und den exakten LilyPond-Code eines Notenbildes sehen wollen, klicken Sie einfach auf das Notenbild. Wenn Sie nicht die HTML-Version lesen, können Sie die angezeigten Code-Beispiele kopieren und in einen Editor einfügen. Sie müssen dabei aber `\relative c' ' { }` einfügen, wie hier gezeigt:

```
\relative c' ' {
  ... hier das Beispiel ...
}
```

Warum werden die Klammern hier meist weggelassen? Die meisten der Beispiele können in ein längeres Musikstück hineinkopiert werden, und dann ist es natürlich nicht sinnvoll, wenn auch noch `\relative c' ' { }` dazukommt; ein `\relative` sollte nicht innerhalb eines anderen `\relative` gesetzt werden, deshalb wird es hier weggelassen, damit die Beispiele auch innerhalb eines anderen Kontextes funktionieren.

2.2 Notation auf einem System

Dieses Kapitel lehrt grundlegende Bestandteile der Notation, die für eine Stimme auf einem System gebraucht werden.

2.2.1 Relative Notenbezeichnungen

Wie wir im Kapitel [Abschnitt 2.1.2 \[Einfache Notation\]](#), Seite 12 gesehen haben, errechnet LilyPond die Tonhöhe der folgenden Noten bezüglich der vorhergehenden.³ Wenn keine zusätzlichen Oktavierungszeichen (' und ,) gesetzt sind, wird angenommen, dass die folgende Tonhöhe sich innerhalb einer Quarte in Bezug auf die vorhergehende befindet.

LilyPond errechnet die Tonhöhen aus den Notenbezeichnungen – eine übermäßige Quarte ist also *nicht* das selbe wie eine verminderte Quinte. Wenn wir von C aus rechnen, wird ein Fis höher als das C gesetzt, während ein Ges in der Quarte *unter* dem C gesetzt wird.

```
c2 fis
```

³ Es gibt noch einen anderen Eingabemodus für Tonhöhen, siehe [Abschnitt 2.5.3 \[Absolute Notenbezeichnungen\]](#), Seite 31, aber für die Praxis ist der relative Modus viel einfacher und sicherer zu benutzen.

c2 ges



Mehr Information

Relativer Oktaveintrag

siehe [Abschnitt 6.1.6 \[Relative Oktaven\]](#), Seite 65.

Oktaveüberprüfung

siehe [Abschnitt 6.1.7 \[Oktavenüberprüfung\]](#), Seite 67.

2.2.2 Versetzungszeichen und Tonartbezeichnung (Vorzeichen)

Versetzungszeichen

Ein Kreuz-Versetzungszeichen⁴ wird eingegeben, indem an den Notennamen ein ‚is‘ gehängt wird, ein B-Vorzeichen durch Anhängen von ‚es‘. Logischerweise wird dann ein **Doppelkreuz** oder **Doppel-B** durch Anhängen von ‚isis‘ oder ‚eses‘ geschrieben.⁵

cis1 ees fisis, aeses



Tonartbezeichnungen (Vorzeichen)

Die Tonart eines Stückes wird erstellt mit dem Befehl `\key`, gefolgt von einer Notenbezeichnung und `\major` (für Dur) oder `\minor` (für Moll).

`\key d \major`

a1

`\key c \minor`

a



Warnung: Tonartbezeichnungen und Tonhöhen

Um zu bestimmen, ob vor einer bestimmten Note ein Versetzungszeichen erscheinen soll, untersucht LilyPond die Notenhöhen und die Tonart. Die Tonart beeinflusst nur die *gedruckten* Versetzungszeichen, nicht die wirklichen Tonhöhen! Diese Besonderheit scheint am Anfang oft verwirrend, so dass sie hier etwas genauer betrachtet wird.

⁴ In der Umgangssprache werden die Versetzungszeichen häufig auch Vorzeichen genannt. In diesem Handbuch wird jedoch zwischen Vorzeichen zur generellen Angabe der Tonart und den Versetzungszeichen, die direkt im Notentext erscheinen, unterschieden.

⁵ Diese Syntax stammt aus der Tradition der germanischen Sprachen und ist also für deutsche Benutzer kein Problem. Es ist aber möglich, die Namen anderer Sprachen zu benutzen, siehe [Abschnitt 6.1.5 \[Notenbezeichnungen in anderen Sprachen\]](#), Seite 65.

LilyPond unterscheidet strikt zwischen dem musikalischen Inhalt und dem Satz (Layout). Die Alteration (Kreuz, Auflösungszeichen oder b) einer Note gehört zur Tonhöhe dazu und ist deshalb musikalischer Inhalt. Ob ein Versetzungszeichen (also ein *gedrucktes* Kreuz, b oder Auflösungszeichen) auch vor der Note erscheint, hängt vom Kontext, also vom Layout ab. Das Layout gehorcht bestimmten Regeln, und Versetzungszeichen werden automatisch nach diesen Regeln gesetzt. Die Versetzungszeichen im fertigen Notenbild sind nach den Regeln des Notensatzes gesetzt. Deshalb wird automatisch entschieden, wo sie erscheinen, und man muss den Ton eingeben, den man *hören* will.

In diesem Beispiel

```
\key d \major
d cis fis
```



hat keine Note ein Versetzungszeichen, trotzdem muss im Quelltext das ‚is‘ für `cis` und `fis` notiert werden.

Der Code ‚e‘ heißt also nicht: „Zeichne einen schwarzen Punkt auf die erste Linie des Systems.“ Im Gegenteil, er heißt vielmehr: „Hier soll eine Note mit der Tonhöhe E gesetzt werden.“ In der Tonart As-Dur *bekommt* sie ein Versetzungszeichen:

```
\key aes \major
e
```



Alle diese Versetzungszeichen ausdrücklich zu schreiben, bedeutet vielleicht etwas mehr Schreibarbeit, hat aber den großen Vorteil, das Transposition sehr viel einfacher gemacht wird und der Druck von Versetzungszeichen nach unterschiedlichen Regeln erfolgen kann. Siehe [Abschnitt 9.1.1 \[Automatische Versetzungszeichen\]](#), [Seite 134](#) für einige Beispiele, wie Vorzeichen anhand von unterschiedlichen Regeln ausgegeben werden können.

Mehr Information

Versetzungszeichen

siehe [Abschnitt 6.1.2 \[Versetzungszeichen\]](#), [Seite 63](#) und [Abschnitt 9.1.1 \[Automatische Versetzungszeichen\]](#), [Seite 134](#).

Tonart (Vorzeichen)

siehe [Abschnitt 6.4.2 \[Tonartbezeichnung\]](#), [Seite 83](#).

2.2.3 Bindebögen und Legaotbögen

Bindebögen

Ein Bindebogen wird geschrieben, indem man eine Tilde ‚~‘ and die erste der zu verbindenden Noten hängt.

```
g4~ g c2~
c4 ~ c8 a8 ~ a2
```



Legatobögen

Ein Legatobogen ist ein Bogen, der sich über mehrere Noten erstreckt. Die Note, an der er beginnt, und die Note, an der er endet, werden mit ‚(‘ beziehungsweise ‚)‘ markiert.

d4(c16) cis(d e c cis d) e(d4)



Phrasierungsbögen

Bögen, die längere Phrasierungseinheiten markieren (Phrasierungsbögen), werden mit ‚\(' und ‚\)' eingegeben. Es können sowohl Legato- als auch Phrasierungsbögen gleichzeitig vorkommen, aber es kann nicht mehr als jeweils einen Legato- und einen Phrasierungsbogen gleichzeitig geben.

a8(\(a1s b c) cis2 b'2 a4 cis,\)



Warnung: Bindebögen sind nicht Legatobögen

Ein Legatobogen sieht aus wie ein Bindebogen, hat aber eine andere Bedeutung. Ein Bindebogen verlängert nur die vorhergehende Note und kann also nur bei zwei Noten gleicher Tonhöhe benutzt werden. Legatobögen dagegen zeigen die Artikulation von Noten an und können für größere Notengruppen gesetzt werden. Binde- und Legatobögen können geschachtelt werden.

c2~(c8 fis fis4 ~ fis2 g2)



Mehr Information

Bindebögen

siehe [Abschnitt 6.5.1 \[Bindebögen\]](#), Seite 92.

Legatobögen

siehe [Abschnitt 6.5.2 \[Legatobögen\]](#), Seite 94.

Phrasierungsbögen

siehe [Abschnitt 6.5.3 \[Phrasierungsbögen\]](#), Seite 95.

2.2.4 Artikulationszeichen und Lautstärke

Artikulationszeichen

Übliche Artikulationszeichen können durch Anfügen von Minus (,-) und einem entsprechenden Zeichen eingegeben werden:

`c- . c-- c-> c-^ c-+ c-_-`



Fingersatz

Auf gleiche Weise können Fingersatzbezeichnungen hinzugefügt werden, indem nach dem Minus (,-) eine Zahl geschrieben wird:

`c-3 e-5 b-2 a-1`



Artikulationszeichen und Fingersätze werden normalerweise automatisch platziert, aber man kann ihre Position auch vorgeben durch die Zeichen ,^ (oben) oder ,_ (unten) anstelle des Minus. An eine Note können auch mehrfache Artikulationszeichen gehängt werden. Meistens findet aber LilyPond alleine die beste Möglichkeit, wie die Artikulationen platziert werden sollen.

`c_-^1 d^ . f^4_2-> e^-_+`



Dynamik

Die Dynamik innerhalb eines Stückes wird eingegeben, indem man die Markierungen (mit einem Backslash, \) an die Note hängt:

`c\ff c\mf c\p c\pp`



Crescendo und Decrescendo werden mit dem Befehl \< beziehungsweise \> begonnen. Ein Dynamik-Zeichen, etwa \f, beendet das (De)Crescendo. Auch mit dem Befehl \! kann es explizit beendet werden.

`c2\< c2\ff\> c2 c2\!`



Mehr Information

Artikulationszeichen

siehe [Abschnitt 6.6.1 \[Artikulationszeichen\]](#), Seite 101.

Fingersatz siehe [Abschnitt 6.6.2 \[Fingersatzanweisungen\]](#), Seite 103.

Dynamik siehe [Abschnitt 6.6.3 \[Dynamik\]](#), Seite 104.

2.2.5 Automatische und manuelle Balken

Alle Balken werden automatisch gesetzt:

```
a8 ais d ees r d c16 b a8
```



Wenn diese automatisch gesetzten Balken nicht gewollt sind, können sie manuell geändert werden. Die Note, an der der Balken anfängt, erhält ein ‚[‘ (AltGr+8) und die, an der er endet, ein ‚]‘ (AltGr+9).

```
a8[ ais] d[ ees r d] a b
```



Mehr Information

Automatische Balken

siehe [Abschnitt 6.5.5 \[Automatische Balken\]](#), Seite 96.

Manuelle Balken

siehe [Abschnitt 6.5.6 \[Manuelle Balken\]](#), Seite 97.

2.2.6 Zusätzliche rhythmische Befehle

Auftakt

Ein Auftakt wird mit dem Befehl `\partial` eingegeben. Darauf folgt die Länge des Auftaktes: `\partial 4` heißt eine Viertelnote Auftakt und `\partial 8` eine Achtelnote.

```
\partial 8  
f8 c2 d
```



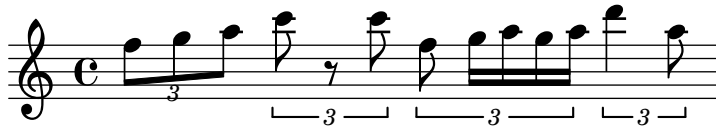
Andere rhythmische Aufteilungen

Triolen und N-tolen werden mit dem `\times`-Befehl erzeugt. Er braucht zwei Argumente: einen Bruch und die Noten, auf die er sich bezieht. Die Dauer des Abschnittes wird mit dem Bruch multipliziert. In einer Triole dauern die Noten $\frac{2}{3}$ ihrer normalen Länge, also hat eine Triole $\frac{2}{3}$ als Bruch:

```

\times 2/3 { f8 g a }
\times 2/3 { c r c }
\times 2/3 { f,8 g16[ a g a] }
\times 2/3 { d4 a8 }

```



Verzierungen

Verzierungen werden mit dem Befehl `\grace` eingegeben, Vorhalte durch den Befehl `\appoggiatura` und Vorschläge mit `\acciaccatura`.

```

c2 \grace { a32[ b] } c2
c2 \appoggiatura b16 c2
c2 \acciaccatura b16 c2

```



Mehr Information

Vorschläge, Verzierungen

siehe [Abschnitt 6.5.7 \[Verzierungen\]](#), Seite 98,

Triolen

siehe [Abschnitt 6.2.3 \[Andere rhythmische Aufteilungen\]](#), Seite 71,

Auftakt

siehe [Abschnitt 6.4.4 \[Auftakte\]](#), Seite 85.

2.3 Mehrere Noten auf einmal

In diesem Kapitel wird gezeigt, wie mehr als eine Note zur gleichen Zeit gesetzt werden kann: auf unterschiedlichen Systemen für verschiedene Instrumente oder für ein Instrument (z. B. Klavier) und in Akkorden.

Polyphonie nennt man in der Musik das Vorkommen von mehr als einer Stimme in einem Stück. Polyphonie bzw. Mehrstimmigkeit heißt für LilyPond allerdings das Vorkommen von mehr als einer Stimme pro System.

2.3.1 Musikalische Ausdrücke erklärt

In LilyPond-Quelldateien wird Musik durch *musikalische Ausdrücke* dargestellt. Eine einzelne Note ist ein musikalischer Ausdruck, auch wenn sie ganz allein ohne Kontext keinen gültigen Code darstellt.

```
a4
```



Eine Gruppe von Noten innerhalb von Klammern bildet einen neuen Ausdruck.

```
{ a4 g4 }
```



Wenn eine Gruppe von musikalischen Ausdrücken (also beispielsweise Noten) in geschweifte Klammern gesetzt wird, bedeutet das, dass eine Gruppe nach der anderen gesetzt wird. Das Resultat ist ein neuer musikalischer Ausdruck.

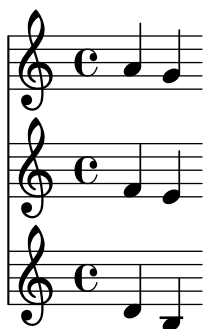
```
{ { a4 g } f g }
```



Gleichzeitige musikalische Ausdrücke: mehrere Notensysteme

Mit dieser Technik kann polyphone Musik gesetzt werden. Musikalische Ausdrücke werden einfach parallel kombiniert, damit sie gleichzeitig als eigene Stimmen in dem gleichen Notensystem gesetzt werden. Um anzuzeigen, dass an dieser Stelle gleichzeitige Noten gesetzt werden, muss nur ein Kombinationszeichen eingefügt werden. Parallel werden musikalische Ausdrücke kombiniert, indem man sie mit << und >> einrahmt. Im folgenden Beispiel sind drei Ausdrücke (jeder mit zwei Noten) parallel kombiniert:

```
\relative c'' {
  <<
    { a4 g }
    { f e }
    { d b }
  >>
}
```



Es ist noch zu bemerken, dass wir hier für jede Ebene innerhalb der Quelldatei eine andere Einrückung geschrieben haben. Für LilyPond spielt es keine Rolle, wieviel Leerzeichen am Anfang einer Zeile sind, aber für Menschen ist es eine große Hilfe, sofort zu sehen, welche Teile des Quelltextes zusammen gehören.

Warnung: Jede Note ist relativ zu der vorhergehenden in der Datei, nicht relativ zu dem zweigestrichenen C (c''), das im `\relative`-Befehl angegeben ist. Die Klammern haben darauf keinen Einfluss.

Gleichzeitige musikalische Ausdrücke: ein Notensystem

Um die Anzahl der Notensysteme zu bestimmen, analysiert LilyPond den ersten Ausdruck. Wenn es sich um eine einzelne Note handelt, wird nur ein System gesetzt, wenn es sich um eine parallele Anordnung von Ausdrücken handelt, wird mehr als ein System gesetzt. Das folgende Beispiel beginnt mit einer Note:

```
\relative c'' {
  c2 <<c e>>
  << { e f } { c <<b d>> } >>
}
```



Analogie: mathematische Ausdrücke

Die Anordnung von Ausdrücken funktioniert ähnlich wie mathematische Gleichungen. Eine längere Gleichung entsteht durch die Kombination kleinerer Gleichungen. Solche Gleichungen werden auch Ausdruck genannt und ihre Definition ist rekursiv, sodass beliebig komplexe und lange Ausdrücke erstellt werden können. So etwa hier:

```
1
1 + 2
(1 + 2) * 3
((1 + 2) * 3) / (4 * 5)
```

Das ist eine Folge von (mathematischen) Ausdrücken, in denen jeder Ausdruck in dem folgenden (größeren) enthalten ist. Die einfachsten Ausdrücke sind Zahlen, und größere werden durch die Kombination von Ausdrücken mit Hilfe von Operatoren (wie '+', '*', und '/') sowie Klammern. Genauso wie mathematische Ausdrücke können auch musikalische Ausdrücke beliebig tief verschachtelt werden. Das wird benötigt für komplexe Musik mit vielen Stimmen.

2.3.2 Mehrere Notensysteme

Wie wir in [Abschnitt 2.3.1 \[Musikalische Ausdrücke erklärt\]](#), [Seite 23](#) gesehen haben, sind LilyPond-Quelldateien aus musikalischen Ausdrücken konstruiert. Wenn die Noteneingabe mit parallelen Ausdrücken beginnt, werden mehrere Notensysteme erstellt. Es ist aber sicherer und einfacher zu verstehen, wenn diese Systeme explizit erstellt werden.

Um mehr als ein System zu schreiben, wird jedem Notenausdruck, der in einem eigenen System stehen soll, der Befehl `\new Staff` vorne angefügt. Diese `Staff` (engl. für Notensystem)-Elemente werden dann parallel angeordnet mit den `<<` und `>>`-Zeichen:

```
\relative c'' {
  <<
    \new Staff { \clef treble c }
    \new Staff { \clef bass c,, }
  >>
}
```



Der Befehl `\new` beginnt einen neuen ‚Notationskontext‘. Ein solcher Notationskontext ist eine Umgebung, in der musikalische Ereignisse (wie Noten oder `\clef` (Schlüssel)-Befehle) interpretiert werden. Für einfache Stücke werden diese Umgebungen automatisch erstellt. Für kompliziertere Musik ist es aber am besten, die Umgebungen explizit zu erstellen.

Es gibt verschiedene Kontext-Typen. **Score** (Partitur), **Staff** (Notensystem) und **Voice** (Stimme) verarbeiten die Eingabe von Noten, während die **Lyrics** (Text)-Umgebung zum Setzen von Liedtexten und die **ChordNames** (Akkordbezeichnungen)-Umgebung für Akkordsymbole verwendet wird.

Die Syntax des `\new`-Befehls erinnert an das Minuszeichen in der Mathematik. Genauso wie $(4+5)$ ein Ausdruck ist, der durch $-(4+5)$ zu einem größeren Ausdruck erweitert wurde, werden auch musikalische Ausdrücke durch den `\new`-Befehl erweitert.

Die Taktangabe, die in einem einzelnen System angegeben wird, wirkt sich auf alle anderen System aus, während die Angabe der Tonart sich nur auf ein einzigen System beschränkt.⁶

```
\relative c'' {
  <<
    \new Staff { \clef treble \time 3/4 c }
    \new Staff { \clef bass \key d \major c,, }
  >>
}
```



2.3.3 Klaviersysteme

Musik für das Klavier wird üblicherweise auf zwei Systemen notiert, die durch eine geschweifte Klammer verbunden sind (Akkolade). Um ein derartiges Notensystem zu erstellen, geht man ähnlich vor wie in dem Beispiel aus [Abschnitt 2.3.2 \[Mehrere Notensysteme\]](#), [Seite 25](#), nur dass der gesamte Ausdruck jetzt in eine **PianoStaff**-Umgebung eingefügt wird.

```
\new PianoStaff <<
  \new Staff ...
  \new Staff ...
>> >>
```

Hier ein kleines Beispiel:

```
\relative c'' {
  \new PianoStaff <<
    \new Staff { \time 2/4 c4 e g g, }
    \new Staff { \clef bass c,, c' e c }
  >>
}
```

⁶ Dies kann natürlich auch geändert werden, siehe [Kapitel 9 \[Standardeinstellungen verändern\]](#), [Seite 134](#) für Einzelheiten.



Mehr Information

Siehe [Abschnitt 7.1 \[Notation für Klavier\]](#), Seite 117.

2.3.4 Mehrstimmigkeit in einem System

Wenn unterschiedliche Melodien oder Stimmen in einem System kombiniert werden sollen, werden sie als „polyphone Stimmen“ realisiert: Jede Stimme hat eigene Hälse, Balken und Legatobögen, und die Hälse der oberen Stimme zeigen immer nach oben, während die Hälse der unteren Stimme nach unten zeigen.

Diese Art von Notenbild wird erstellt, indem jede Stimme für sich als Abfolge notiert wird (mit {...}) und diese dann parallel kombiniert werden, indem \\ zwischen die einzelnen Stimmen gesetzt wird.

```
<<
  { a4 g2 f4~ f4 } \\
  { r4 g4 f2 f4 }
>>
```



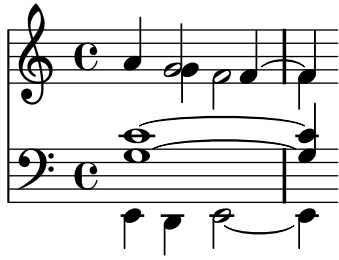
Für den Satz von mehrstimmigen Stücken kann es auch angebracht sein, unsichtbare Pausen zu verwenden. Hiermit könne Stimmen ausgefüllt werden, die gerade nicht aktiv sind. Hier ist ein Beispiel mit einer unsichtbaren Pause (,s') anstelle einer normalen (,r')

```
<<
  { a4 g2 f4~ f4 } \\
  { s4 g4 f2 f4 }
>>
```



Auch diese Ausdrücke wiederum könne beliebig miteinander kombiniert werden.

```
<<
  \new Staff <<
    { a4 g2 f4~ f4 } \\
    { s4 g4 f2 f4 }
  >>
  \new Staff <<
    \clef bass
    { <c g>1 ~ <c g>4 } \\
    { e,,4 d e2 ~ e4}
  >>
>>
```



Mehr Information

Siehe [Abschnitt 6.3.3 \[Einfache Mehrstimmigkeit\]](#), Seite 76.

2.3.5 Noten zu Akkorden verbinden

Akkorde werden notiert, indem die Tonhöhen von spitzen Klammern (`<`, `<'` und `>'`) umgeben werden.

```
r4 <c e g>4 <c f a>2
```

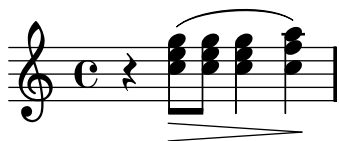


Auch andere Markierungen wie Balken oder Bögen können mit den Akkorden kombiniert werden. Sie müssen jedoch außerhalb der spitzen Klammern gesetzt werden.

```
r4 <c e g>8[ <c f a>]~ <c f a>2
```



```
r4 <c e g>8\>( <c e g> <c e g>4 <c f a>\!)
```



2.4 Lieder

In diesem Kapitel wird in die Kombination von Musik mit Text eingeführt und die Erstellung einfacher Song-Blätter gezeigt.

2.4.1 Setzen von Text

Hier haben wir eine einfache Melodie:

```
\relative c'' {
  a4 e c8 e r4
  b2 c4( d)
}
```



Zu diesen Noten kann Text hinzugefügt werden, indem beide mit dem `\addlyrics`-Befehl kombiniert werden. Text wird eingegeben, indem jede Silbe durch ein Leerzeichen getrennt wird.

```
<<
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free }
>>
```



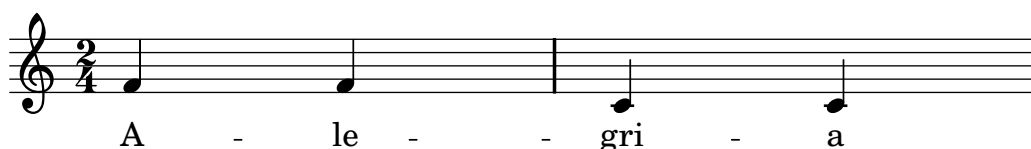
Diese Melodie endet in einem *Melisma*, d. h. eine einzige Silbe wird auf mehr als einer Note gesungen. Das wird im Text meistens mit einem Unterstrich hinter der Silbe dargestellt. Dieser Strich wird durch zwei Unterstriche (`--`), von Leerzeichen umgeben, notiert.

```
<<
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free -- }
>>
```



Auf gleiche Art können Trennstriche zwischen einzelnen Silben eines Wortes mit zwei Minuszeichen (`--`) eingegeben werden. Sie werden im Notenbild als eine oder mehrere zentrierte Trennstriche dargestellt.

```
<<
  \relative c' {
    \time 2/4
    f4 f c c
  }
  \addlyrics { A -- le -- gri -- a }
>>
```



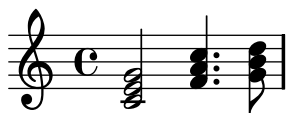
Mehr Information

Mehr Möglichkeiten, wie etwa mehrere Strophen unter der gleichen Melodie, werden im Kapitel [Abschnitt 7.3 \[Notation von Gesang\]](#), Seite 117 gezeigt.

2.4.2 Ein Song-Blatt

In der Pop-Musik wird die Begleitung gerne mit Akkordsymbolen angegeben. Diese Bezeichnungen können in der `\chordmode`-Umgebung wie Noten eingegeben werden:

```
\chordmode { c2 f4. g8 }
```



Jede Tonhöhe wird als die Basis eines Akkordes interpretiert. Andere Akkorde als die Grundakkorde können durch Anhängen besonderer Bezeichnungen erstellt werden. Diese Bezeichnungen müssen an den Tonhöhenbuchstaben gehängt werden, getrennt von einem Doppelpunkt. Im nächsten Beispiel werden einige der gängigsten Bezeichnungen dargestellt:

```
\chordmode { c2 f4:m g4:maj7 gis1:dim7 }
```



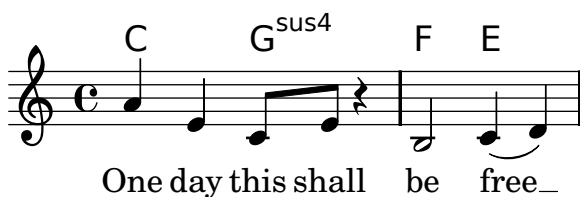
Für Liederblätter werden die Akkorde jedoch meistens nicht als Noten dargestellt, sondern als Symbole über dem Text. Das erreicht man durch die Umgebung `\chord` anstelle des eben verwendeten `\chordmode`. Es kann die gleiche Syntax verwendet werden, aber die Tonbezeichnungen ergeben jetzt Akkordsymbole:

```
\chords { c2 f4.:m g4.:maj7 gis8:dim7 }
```

C FmG[△] G^{#07}

Indem man jetzt die Akkordsymbole, den Text und eine Melodie kombiniert, hat man ein Song-Blatt.

```
<<
\chords { c2 g:sus4 f e }
\relative c'' {
  a4 e c8 e r4
  b2 c4( d)
}
\addlyrics { One day this shall be free __ }
>>
```



Mehr Information

Eine vollständige Liste der Akkordbezeichnungen und andere Optionen für das Layout sind im Kapitel [Abschnitt 6.3.1 \[Akkorde\]](#), [Seite 75](#) zu finden.

2.5 Letzter Schliff

Das ist das letzte Kapitel der Übung. Hier soll demonstriert werden, wie man den letzten Schliff an einfachen Stücken anbringen kann. Gleichzeitig dient es als Einleitung zum Rest des Handbuches.

2.5.1 Versionsnummer

Der `\version`-Befehl zeigt an, für welche LilyPond-Version eine Quelldatei geschrieben worden ist. Um etwa eine Datei für die Version 2.11.20 zu markieren, wird einfach

```
\version "2.11.15"
```

am Anfang der Textdatei eingefügt.

Durch diese Versionsmarkierung werden zukünftige Aktualisierungen des LilyPond-Programmes einfacher gemacht. Syntax-Änderungen zwischen den Programmversionen werden von einem speziellen Programm, `'convert-ly'`, vorgenommen (siehe `program usage manual`, `Updating files with convert-ly`). Dieses Programm braucht `\version`, um zu entscheiden, welche Regeln angewandt werden müssen.

2.5.2 Titel hinzufügen

Titel, Komponist, Opusnummern und ähnliche Information werden in einer `\header`-Umgebung eingefügt. Diese Umgebung befindet sich außerhalb der musikalischen Ausdrücke, meistens wird die `\header`-Umgebung direkt nach der Versionsnummer eingefügt.

```
\version "2.11.15"
\header {
  title = "Symphony"
  composer = "Ich"
  opus = "Op. 9"
}

{
  ... Noten ...
}
```

Wenn die Datei übersetzt wird, werden Titel- und Komponisteneinträge über der Musik ausgegeben. Mehr Information über die Titelei findet sich im Kapitel [Abschnitt 10.2.1 \[Titel erstellen\]](#), Seite 136.

2.5.3 Absolute Notenbezeichnungen

Bis jetzt haben wir immer `\relative` benutzt, um Tonhöhen zu bestimmen. Das ist die einfachste Eingabeweise für die meiste Musik. Es gibt aber noch eine andere Möglichkeit, Tonhöhen darzustellen: durch absolute Bezeichnung.

Wenn man das `\relative` weglässt, werden alle Tonhöhen von LilyPond als absolute Werte interpretiert. Ein `c'` ist dann also immer das eingestrichene C, ein `b` ist immer das kleine h unter dem eingestrichenen C, und ein `g`, ist immer das große G – also die Note auf der letzten Linie im Bass-Schlüssel.

```
{
  \clef bass
  c' b g, g,
  g, f, f c'
}
```



Hier eine Tonleiter über vier Oktaven:

```
{
  \clef bass
  c, d, e, f,
  g, a, b, c
  d e f g
  a b c' d'
  \clef treble
  e' f' g' a'
  b' c'' d'' e''
  f'' g'' a'' b''
  c'''1
}
```



Wie leicht zu sehen ist, muss man sehr viele Apostrophe schreiben, wenn die Melodie im Sopranschlüssel notiert ist. Siehe etwa dieses Fragment von Mozart:

```
{
  \key a \major
  \time 6/8
  cis''8. d''16 cis''8 e''4 e''8
  b'8. cis''16 b'8 d''4 d''8
}
```



Alle diese Apostrophe machen den Quelltext schlecht lesbar und sind eine mögliche Fehlerquelle. Mit dem `\relative`-Befehl ist das Beispiel sehr viel einfacher zu lesen:

```
\relative c'' {
  \key a \major
  \time 6/8
  cis8. d16 cis8 e4 e8
  b8. cis16 b8 d4 d8
}
```



Wenn man einen Fehler durch ein Oktavierungszeichen (' oder ,) im `\relative`-Modus macht, ist er sehr schnell zu finden, denn viele Noten sind nacheinander in der falschen Oktave. Im absoluten Modus dagegen ist ein einzelner Fehler nicht so deutlich und deshalb auch nicht so einfach zu finden.

Trotz allem ist der absolute Modus gut für Musik mit sehr großen Sprüngen und vor allem für computergenerierte LilyPond-Dateien.

2.5.4 Stücke durch Bezeichner organisieren

Wenn alle die Elemente, die angesprochen wurden, zu größeren Dateien zusammengefügt werden, werden auch die musikalischen Ausdrücke sehr viel größer. In polyphonen Dateien mit vielen Systemen kann das sehr chaotisch aussehen. Das Chaos kann aber deutlich reduziert werden, wenn **Bezeichner** definiert und verwendet werden.

Bezeichner (die auch als Variablen oder Makros bezeichnet werden) können einen Teil der Musik aufnehmen. Sie werden wie folgt definiert:

```
bezeichneteMusik = { ... }
```

Der Inhalt des musikalischen Ausdrucks `bezeichneteMusik` kann dann später wieder benutzt werden, indem man einen Backslash davor setzt (`\bezeichneteMusik`), genau wie bei jedem LilyPond-Befehl. Bezeichner müssen *vor* dem eigentlichen musikalischen Ausdruck definiert werden.

```
violin = \new Staff { \relative c' {
  a4 b c b
}}
cello = \new Staff { \relative c {
  \clef bass
  e2 d
}}
{
  <<
    \violin
    \cello
  >>
}
```



In den Namen der Bezeichner dürfen nur Buchstaben des Alphabets verwendet werden, keine Zahlen oder Striche.

Man kann diese Variablen auch für alle anderen Objekte verwenden, etwa:

```
width = 4.5\cm
Name = "Wendy"
aFünfPapier = \paper { paperheight = 21.0 \cm }
```

Abhängig vom Kontext kann solch ein Bezeichner in verschiedenen Stellen verwendet werden. Das folgende Beispiel zeigt die Benutzung der eben definierten Bezeichner:

```
\paper {
  \aFünfPapier
```

```

    line-width = \width
}
{ c4^\Name }

```

2.5.5 Nach der Übung

Wenn Sie diese Übung absolviert haben, sollten Sie am besten ein paar Stücke selber notieren. Beginnen Sie mit den [Anhang D \[Vorlagen\]](#), [Seite 148](#) und fügen Sie einfach Ihre Noten dazu. Wenn Sie irgendetwas brauchen, das nicht in der Übung besprochen wurde, schauen Sie sich den Abschnitt Alles über die Notation an, angefangen mit [Kapitel 6 \[Grundlegende Notation\]](#), [Seite 62](#). Wenn Sie für ein Instrument oder Ensemble Noten schreiben wollen, für das es keine Vorlage gibt, schauen Sie ins Kapitel [Abschnitt 3.1 \[Erweiterung der Beispiele\]](#), [Seite 35](#).

Wenn Sie ein paar kurze Stücke notiert haben, lesen Sie den Rest des Lernhandbuchs (Kapitel 3–5). Natürlich können Sie auch sofort weiterlesen. Die nächsten Kapitel sind aber mit der Annahme geschrieben, dass Sie die Eingabesprache von LilyPond beherrschen. Sie können die weiteren Kapitel auch überfliegen und dann darauf wieder zurückkommen, wenn Sie einige Erfahrung im Notieren gewonnen haben.

2.5.6 Wie soll das Handbuch gelesen werden

Wie wir im Kapitel [Abschnitt 2.1.4 \[Wie soll man die Übung lesen\]](#), [Seite 17](#) gesehen haben, haben wir bei vielen Beispielen in der Übung die `\relative c''`-Umgebung weggelassen.

Im Rest des Handbuchs wurde noch viel nachlässiger vorgegangen: In manchen Fällen fehlt nur das `\relative c''`, in anderen wurde eine andere Oktave als Startpunkt benutzt (etwa `c'` oder `c,,`), manchmal ist das ganze Beispiel im absoluten Modus geschrieben. Solche Mehrdeutigkeiten treten aber nur in Fällen auf, in denen die Tonhöhe nicht wichtig ist. In allen Beispielen, in denen die Tonhöhe eine Rolle spielt, sind entweder `\relative` oder die Klammern `{ }` für den absoluten Modus vermerkt.

Wenn Sie noch nicht genau verstanden haben, wie eigentlich der Quellcode genau aussehen soll, schauen Sie sich am besten die HTML-Version dieses Dokuments an (wenn Sie das nicht schon tun) und klicken Sie auf die Notenbilder, um die dazugehörige Textdatei anzuschauen, mit der dieses Notenbild erstellt worden ist.

3 Alles zusammenfügen

Dieses Kapitel behandelt das allgemeine Konzept von LilyPond und wie man `\score`-Blöcke erstellt.

3.1 Erweiterung der Beispiele

Wenn Sie das Übungskapitel gelesen haben, wissen Sie, wie man Noten schreibt. Aber wie erhalten Sie genau die Notensysteme, die Sie sich vorstellen? Die Vorlagen sind gut, aber was, wenn Sie etwas anderen wollen?

Beginnen Sie mit der Vorlage, die am ehesten dem nahe kommt, was Sie als Ergebnis haben wollen. Sagen wir, Sie wollen ein Stück für Sopran und Cello notieren. In diesem Fall beginnen Sie mit „Noten und Text“ (für die Sopranstimme).

```
\version "2.11.15"
melodie = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

\score{
  <<
    \new Voice = "one" {
      \autoBeamOff
      \melodie
    }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}
```

Jetzt wollen wir die Cellostimme hinzufügen. Schauen wir uns das „Nur Noten“-Beispiel an:

```
\version "2.11.15"
melodie = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

\score {
  \new Staff \melodie
  \layout { }
  \midi { }
```

```
}
```

Wir brauchen nicht zwei `\version`-Befehle. Wir brauchen den `melodie`-Abschnitt. Wir brauchen keine zwei `\score`-Abschnitte, denn damit würden wir ja zwei voneinander getrennte Notensysteme erhalten. Wir wollen sie aber zusammen, als Duo. Innerhalb eines `\score`-Abschnittes brauchen wir auch nicht zweimal `\layout` und `\midi`.

Kopieren wir aber nur den `melodie`-Abschnitt, hätte wir zwei `melodie`-Abschnitte. Also nennen wir diesen Bezeichner anders. Die Noten für den Sopran nennen wir `sopranNoten`, die Noten für das Cello `celloNoten`. Gleichzeitig können wir auch noch den Abschnitt `text` in `sopranText` umbenennen. Es ist wichtig, beide Befehle umzubenennen, einmal die Definition (also den `melodie = \relative c' { -Teil}`), zweitens die Anwendung des Namens in der Partitur (also in dem `\score`-Abschnitt).

Während wir hier diese Änderungen vornehmen, können wir auch gleich das Cello-System anpassen – Celli brauchen im Normalfall den Bass-Schlüssel. Wir setzen hier auch ein paar andere Noten für das Cello.

```
\version "2.11.15"
sopranNoten = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

sopranText = \lyricmode {
  Aaa Bee Cee Dee
}

celloNoten = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  d4 g fis8 e d4
}

\score{
  <<
    \new Voice = "one" {
      \autoBeamOff
      \sopranoNoten
    }
    \new Lyrics \lyricsto "one" \sopranText
  >>
  \layout { }
  \midi { }
}
```

Das sieht schon ganz gut aus, aber das Cello-System erscheint noch nicht in der Partitur. Wir haben den Bezeichner ja auch nicht im `\score`-Abschnitt benutzt. Wenn wir wollen, dass die Noten auch gedruckt werden, müssen wir noch

```
\new Staff \celloNoten
```

unter dem Sopran-System einfügen. Wir müssen auch die Zeichen << und unter der Musik dann die Zeichen >> einfügen, denn dadurch wird LilyPond mitgeteilt, dass mehrere Ausdrücke (hier unsere zwei Systeme (`Staff`) gleichzeitig gesetzt werden sollen. Der `\score`-Abschnitt sieht jetzt so aus:

```
\score{
  <<
    <<
      \new Voice = "one" {
        \autoBeamOff
        \sopranNoten
      }
      \new Lyrics \lyricsto "one" \sopranText
    >>
    \new Staff \celloNoten
  >>
  \layout { }
  \midi { }
}
```

Das sieht noch etwas durcheinander aus, die Einzüge sind in Unordnung geraten. Aber das ist einfach wieder hergestellt. Hier die vollständige Vorlage für Sopran und Cello:

```
\version "2.11.15"
sopranoMusic = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

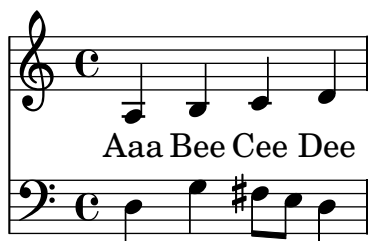
sopranoLyrics = \lyricmode {
  Aaa Bee Cee Dee
}

celloMusic = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  d4 g fis8 e d4
}

\score{
  <<
    <<
      \new Voice = "one" {
        \autoBeamOff
        \sopranoMusic
      }
      \new Lyrics \lyricsto "one" \sopranoLyrics
    >>
    \new Staff \celloMusic
```

```
>>
\layout { }
\midi { }
}
```



3.2 Wie eine LilyPond-Datei funktioniert

Das LilyPond Eingabeformat hat eine ziemlich freie Form, so dass für erfahrene Benutzer viel Freiheit besteht, die Struktur ihrer Quelldateien anzulegen. Für Neulinge kann diese Flexibilität aber erst einmal verwirrend sein. In diesem Kapitel soll darum ein Teil dieser Strukturen dargestellt werden, vieles aber zur Vereinfachung auch weggelassen werden. Für eine komplette Beschreibung des Eingabeformats siehe [Abschnitt 10.1.2 \[Die Dateistruktur\]](#), Seite 136.

Die meisten Beispiele in diesem Handbuch sind kleine Schnipsel, wie etwa dieser:

```
c4 a b c
```

Wie hoffentlich bekannt ist, lässt sich solch ein Schnipsel nicht in dieser Form übersetzen. Diese Beispiele sind also nur Kurzformen von wirklichen Beispielen. Sie müssen wenigstens zusätzlich in geschweifte Klammern gesetzt werden.

```
{
  c4 a b c
}
```

Die meisten Beispiele benutzen auch den `\relative c''`-Befehl. Der ist nicht nötig, um die Dateien zu übersetzen, aber in den meisten Fällen sieht der Notensatz seltsam aus, wenn man den Befehl weglässt.

```
\relative c'' {
  c4 a b c
}
```



Jetzt kommt noch eine Ebene dazu: LilyPond-Code in der obigen Form ist in Wirklichkeit auch wieder eine Abkürzung. Auch wenn man so Dateien schreiben kann und sie auch korrekt gesetzt werden, heißt der wirkliche Code, der hier gemeint ist, eigentlich:

```
\score {
  \relative c'' {
    c4 a b c
  }
}
```

Eine Partitur (`\score`) muss immer mit einem musikalischen Ausdruck beginnen. Das ist letztendlich alle Musik, angefangen bei einer einzelnen Note bis hin zu einer riesigen Partitur mit vielen Notensystemen (bezeichnet durch `GrandStaff`):

```

{
  \new GrandStaff <<
    hier die gesamte Partitur
  >>
}

```

Da sich alles innerhalb der geschweiften Klammern `{ ... }` befindet, wird es wie ein einziger musikalischer Ausdruck behandelt.

Ein `\score` auch andere Dinge enthalten, wie etwa

```

\score {
  { c'4 a b c' }
  \layout { }
  \midi { }
  \header { }
}

```

Viele setzen einige dieser Befehle außerhalb des `\score`-Blocks, zum Beispiel wird der `\header` sehr oft oberhalb der `\score`-Umgebung gesetzt. Das funktioniert genauso gut.

Eine gute Möglichkeit zur Vereinfachung sind selbst definierte Variablen. Alle Vorlagen verwenden diese Möglichkeit.

```

melodie = \relative c' {
  c4 a b c
}

\score {
  { \melodie }
}

```

Wenn LilyPond diese Datei analysiert, nimmt es den Inhalt von `melodie` (alles, was nach dem Gleichheitszeichen kommt) und fügt ihn immer dann ein, wenn ein `\melodie` vorkommt. Die Namen sind frei wählbar, die Variable kann genauso gut `melodie`, `GLOBAL`, `rechteHandklavier`, oder `foofoobarbaz` heißen. Für mehr Information siehe [Abschnitt 4.2 \[Tipparbeit sparen durch Bezeichner und Funktionen\]](#), Seite 46.

Eine komplette Definition des Eingabeformats findet sich im Kapitel [Abschnitt 10.1.2 \[Die Dateistruktur\]](#), Seite 136.

3.3 Score ist ein einziger musikalischer Ausdruck

Im vorigen Kapitel, [Abschnitt 3.2 \[Wie eine LilyPond-Datei funktioniert\]](#), Seite 38, wurde die allgemeine Struktur einer LilyPond-Quelldatei beschrieben. Aber anscheinend haben wir die wichtigste Frage ausgelassen, nämlich wie man herausfindet, was nach dem `\score` geschrieben werden soll.

In Wirklichkeit ist das aber gar kein Geheimnis. Diese Zeile ist die Antwort:

Eine Partitur fängt immer mit `\score` an, gefolgt von einem einzelnen musikalischen Ausdruck.

Vielleicht wollen Sie noch einmal [Abschnitt 2.3.1 \[Musikalische Ausdrücke erklärt\]](#), Seite 23 überfliegen. In diesem Kapitel wurde gezeigt, wie sich große musikalische Ausdrücke aus kleinen Teilen zusammensetzen. Noten können zu Akkorden verbunden werden usw. Jetzt gehen wir aber in die andere Richtung und betrachten, wie sich ein großer musikalischer Ausdruck zerlegen lässt.

```

\score {
  { % diese Klammer startet den großen mus. Ausdruck
    \new GrandStaff <<

```

```

        hier eine ganze Wagner-Oper einfügen
    >>
} % diese Klammer beendet den Ausdruck
\layout { }
}

```

Eine Wagner-Oper ist mindestens doppelt so lang wie dieses Handbuch, beschränken wir uns also auf einen Sänger und Klavier. Wir brauchen keine Orchesterpartitur (**GrandStaff**) dafür, darum lassen wir den Befehl weg. Wir brauchen aber einen Sänger und ein Klavier.

```

\score {
{
<<
\new Staff = "Sänger" <<
>>
\new PianoStaff = Klavier <<
>>
>>
}
\layout { }
}

```

Zur Erinnerung: mit << und >> werden Noten gleichzeitig gesetzt; wir wollen ja auch Klavier- und Sängerstimme gleichzeitig haben.

```

\score {
{
<<
\new Staff = "Sänger" <<
\new Voice = "vocal" { }
>>
\new Lyrics \lyricsto vocal \new Lyrics { }
\new PianoStaff = "piano" <<
\new Staff = "upper" { }
\new Staff = "lower" { }
>>
>>
}
\layout { }
}

```

Jetzt haben wir viel mehr Details. Wir haben ein System (engl. *staff*) für einen Sänger, in dem sich wieder eine Stimme (engl. *voice*) befindet. **Voice** bedeutet für LilyPond eine Stimme (sowohl gesungen als auch gespielt) und evtl. zusätzlich einen Text. Zusätzlich werden zwei Notensysteme für das Klavier mit dem Befehl **\new PianoStaff** gesetzt. **PianoStaff** bezeichnet die Piano-Umgebung (etwa durchgehende Taktstriche und die geschweifte Klammer am Anfang), in der dann wiederum zwei eigene Systeme ("upper" für die rechte Hand und "lower" für die linke) erstellt werden.

Jetzt könnte man in diese Umgebung Noten einfügen. Innerhalb der geschweiften Klammern neben **\new Voice = vocal** könnte man

```

\relative c'' {
a4 b c d
}

```

schreiben. Aber wenn man seine Datei so direkt schreibt, wird der `\score`-Abschnitt sehr lang und es wird ziemlich schwer zu verstehen, wie alles zusammenhängt. Darum bietet es sich an, Bezeichner (oder Variablen) zu verwenden.

```
melodie = { }
text = { }
upper = { }
lower = { }
\score {
  {
    <<
    \new Staff = "Sänger" <<
    \new Voice = "vocal" { \melodie }
    >>
    \new Lyrics \lyricsto vocal \new Lyrics { \text }
    \new PianoStaff = "piano" <<
    \new Staff = "upper" { \upper }
    \new Staff = "lower" { \lower }
    >>
  }
  \layout { }
}
```

Nochmal: der Bezeichner kann aller möglicher Text sein. Die Einschränkungen sind in [Abschnitt 10.1.2 \[Die Dateistruktur\], Seite 136](#) genau aufgelistet.

Beim Schreiben (oder Lesen) einer `\score`-Umgebung sollte man langsam und sorgfältig vorgehen. Am besten fängt man mit dem größten Gebilde an und definiert dann die darin enthaltenen kleineren der Reihe nach. Es hilft auch, sehr genau mit den Einzügen zu sein, so dass jede Zeile, die der gleichen Ebene angehört, wirklich horizontal an der gleichen Stelle beginnt.

3.4 Eine Orchesterstimme

Orchesternoten werden alle zweimal gesetzt. Erstens als Stimmen für die Musiker, und dann als große Partitur für den Dirigenten. Mit Variablen kann hier doppelte Arbeit erspart werden. Die Musik muss nur einmal eingegeben werden und wird in einer Variable abgelegt. Der Inhalt dieser Variable wird dann benutzt, um sowohl die Stimme als auch die Partitur zu erstellen.

Es bietet sich an, die Noten in eigenen Dateien zu speichern. Sagen wir beispielsweise, dass in der Datei `'Horn-Noten.ly'` die folgenden Noten eines Duetts für Horn und Fagott gespeichert sind:

```
HornNoten = \relative c {
  \time 2/4
  r4 f8 a cis4 f e d
}
```

Daraus wird dann eine eigene Stimme gemacht, indem folgende Datei erstellt wird:

```
\include "Horn-Noten.ly"
\header {
  instrument = "Horn in F"
}

{
  \transpose f c' \HornNoten
```

}

Die Zeile

`\include "Horn-Noten.ly"`

setzt den Inhalt der Datei ‘Horn-Noten.ly’ an die Stelle des Befehls in die aktuelle Datei. Damit besteht also eine Definition für `HornNoten`, so dass die Variable verwendet werden kann. Der Befehl `\transpose f c'` zeigt an, dass das Argument, also `\HornNoten`, um eine Quinte nach oben transponiert wird. Klingendes ‘f’ wird also als ‘c’ notiert. Das entspricht der Notation eines Waldhorn in F. Die Transposition zeigt die folgende Ausgabe:



In Musik für mehrere Instrumente kommt es oft vor, dass eine Stimme für mehrere Takte nicht spielt. Das wird mit einer besonderen Pause angezeigt, dem Pausenzeichen für mehrere Takte (engl. multi-measure rest). Sie wird mit dem *großen* Buchstaben ‘R’ eingegeben, gefolgt von einer Dauer (1 für eine Ganze, 2 für eine Halbe usw.). Indem man die Dauer multipliziert, können längere Pausen erstellt werden. Z. B. dauert diese Pause drei Takte eines 2/4-Taktes:

`R2*3`

Wenn die Stimme gedruckt wird, müssen diese Pausen zusammengezogen werden. Das wird durch eine Variable erreicht:

`\set Score.skipBars = ##t`

Dieser Befehl setzt die Eigenschaft des `skipBars` („überspringe Takte“) auf wahr (`##t`). Wenn diese Option und die Pause zu der Musik des Beispiels gesetzt wird, erhält man folgendes Ergebnis:

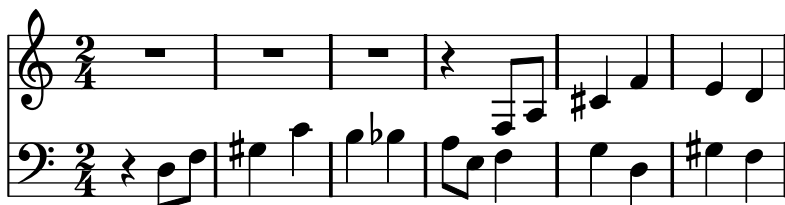


Die Partitur wird erstellt, indem alle Noten zusammengesetzt werden. Angenommen, die andere Stimme trägt den Namen `FagottNoten` und ist in der Datei ‘Fagott-Noten.ly’ gespeichert. Die Partitur sieht dann folgendermaßen aus:

```
\include "Fagott-Noten.ly"
\include "Horn-Noten.ly"
```

```
<<
  \new Staff \HornNoten
  \new Staff \FagottNoten
>>
```

Und mit LilyPond übersetzt:



Tiefer gehende Information darüber, wie Stimmauszüge und Partituren erstellt werden, finden sich im Notationshandbuch, siehe [Abschnitt 8.3 \[Orchestermusik\]](#), [Seite 131](#).

Das Setzen der Variablen, die das Verhalten von LilyPond beeinflussen („properties“), wird im Kapitel [Abschnitt 9.2.3 \[Umgebungseigenschaften lokal ändern\]](#), [Seite 134](#) besprochen.

4 An LilyPond-Projekten arbeiten

Dieses Kapitel erklärt, wie bestimmte häufige Probleme zu lösen oder ganz zu vermeiden sind. Wenn Sie schon Programmiererfahrung mitbringen, erscheinen diese Hinweise vielleicht überflüssig, aber es wird dennoch empfohlen, dieses Kapitel zu lesen.

4.1 Vorschläge, wie LilyPond-Dateien geschrieben werden sollen

Jetzt sind Sie so weit, größere Stücke mit LilyPond zu schreiben – nicht nur die kleinen Beispiele aus der Übung, sondern ganze Stücke. Aber wie geht man das am besten an?

Solange LilyPond Ihre Dateien versteht und die Noten so setzen, wie Sie das wollen, spielt es eigentlich keine Rolle, wie Ihre Dateien aussehen. Es gibt aber trotzdem ein paar Dinge, die man beim Schreiben von LilyPond-Code berücksichtigen sollte.

- Was ist, wenn Sie einen Fehler machen? Die Struktur einer LilyPond-Datei kann es erleichtern (oder erschweren), bestimmte Fehler zu finden.
- Was ist, wenn Sie ihre Dateien mit jemandem austauschen wollen? Oder ihre Dateien nach einige Jahren noch einmal überarbeiten wollen? Manche LilyPond-Dateien versteht man auf den ersten Blick, über anderen muss man eine Stunde grübeln, um die Struktur zu ahnen.
- Was ist, wenn sie Ihre Dateien auf eine neuere LilyPond-Version aktualisieren wollen? Die Syntax der Eingabesprache verändert sich allmählich mit Verbesserungen im Programm. Die meisten Veränderungen können automatisch durch `convert-ly` gelöst werden, aber bestimmte Änderungen brauchen Handarbeit. LilyPond-Dateien können strukturiert werden, damit sie einfacher aktualisierbar sind.

4.1.1 Allgemeine Vorschläge

Hier einige Vorschläge wie Sie Probleme vermeiden oder lösen können:

- **Schreiben Sie immer mit `\version` die Versionsnummer in jede Datei.** Beachten Sie, dass in allen Vorlagen die Versionsnummer `\version "2.11.15"` eingetragen ist. Es empfiehlt sich, in alle Dateien, unabhängig von ihrer Größe, den `\version`-Befehl einzufügen. Persönliche Erfahrung hat gezeigt, dass es ziemlich frustrierend sein kann zu erinnern, welche Programmversion man etwa vor einem Jahr verwendet hat. Auch `convert-ly` benötigt die Versionsnummer.
- **Benutzen Sie Überprüfungen:** [Abschnitt 6.2.5 \[Taktüberprüfung\]](#), [Seite 74](#), [Abschnitt 6.1.7 \[Oktavenüberprüfung\]](#), [Seite 67](#) und [Abschnitt 6.2.6 \[Taktnummerüberprüfung\]](#), [Seite 74](#). Wenn Sie hier und da diese Überprüfungen einfügen, finden Sie einen möglichen Fehler weit schneller. Wie oft aber ist „hier und da“? Das hängt von der Komplexität der Musik ab. Bei einfachen Stücken reicht es vielleicht ein- oder zweimal, in sehr komplexer Musik sollte man sie vielleicht in jeden Takt einfügen.
- **Ein Takt pro Textzeile.** Wenn irgendetwas kompliziertes vorkommt, entweder in der Musik selber oder in der Anpassung der Ausgabe, empfiehlt es sich oft, nur einen Takt pro Zeile zu schreiben. Bildschirmplatz zu sparen, indem Sie acht Takte in eine Zeile zwingen, hilft nicht weiter, wenn Sie ihre Datei „debuggen“ müssen.
- **Kommentieren Sie ihre Dateien.** Benutzen Sie entweder Taktnummern (in regelmäßigen Abständen) oder Verweise auf musikalische Themen („Zweites Thema in den Geigen“, „vierte Variation“ usw.). Sie brauchen diese Kommentare vielleicht noch nicht, wenn Sie das Stück notieren, aber spätestens wenn Sie nach ein paar Jahren etwas verändern wollen oder Sie den Quelltext an einen Freund weitergeben wollen, ist es weitaus komplizierter, die Dateistruktur ohne Kommentare zu verstehen als wenn Sie sie rechtzeitig eingefügt hätten.
- **Schreiben Sie Klammern mit Einrückung.** Viele Probleme entstehen durch ungerade Anzahl von `{` and `}`-Klammern.

- **Schreiben Sie Tondauerangaben** am Anfang von Abschnitten und Bezeichnern. Wenn Sie beispielsweise `c4 d e` am Anfang eines Abschnittes schreiben, ersparen Sie sich viele Probleme, wenn Sie ihre Musik eines Tages umarrangieren wollen.
- **Trennen Sie Einstellungen** von den eigentlichen Noten. Siehe auch [Abschnitt 4.2 \[Tipparbeit sparen durch Bezeichner und Funktionen\]](#), Seite 46 und [Abschnitt 4.3 \[Stil-Dateien\]](#), Seite 48.

4.1.2 Das Kopieren von existierender Musik

Wenn Sie Musik aus einer fertigen Partitur kopieren (z. B. die LilyPond-Eingabe einer gedruckten Partitur):

- Schreiben Sie ein System ihrer Quelle nach dem anderen (aber trotzdem nur einen Takt pro Textzeile) und überprüfen Sie jedes System, nachdem Sie es fertig kopiert haben. Mit dem `showLastLength`-Befehl können Sie den Übersetzungsprozess beschleunigen. Siehe auch [Abschnitt 10.5 \[Korrigierte Musik überspringen\]](#), Seite 137.
- Definieren Sie `mBreak = { \break }` schreiben Sie `\mBreak` in der Quelldatei immer dann, wenn im Manuskript ein Zeilenumbruch vorkommt. Das macht es einfacher, die gesetzte Zeile mit den ursprünglichen Noten zu vergleichen. Wenn Sie die Partitur fertig gestellt haben, könne Sie `mBreak = { }`, also leer definieren, um diese manuellen Zeilenumbrüche zu entfernen. Damit kann dann LilyPond selber entscheiden, wohin es passende Zeilenumbrüche plazierte.

4.1.3 Große Projekte

Besonders wenn sie an größeren Projekten arbeiten, ist es unumgänglich, dass Sie ihre LilyPond-Dateien klar strukturieren.

- **Verwenden Sie Variablen für jede Stimme**, innerhalb der Definition sollte sowenig Struktur wie möglich sein. Die Struktur des `\score`-Abschnittes verändert sich am ehesten, während die `violin`-Definition sich wahrscheinlich mit einer neuen Programmversion nicht verändern wird.

```
violin = \relative c' {
  g4 c'8. e16
}
...
\score {
  \new GrandStaff {
    \new Staff {
      \violin
    }
  }
}
```

- **Trennen Sie Einstellungen von den Noten**. Diese Empfehlung wurde schon im Kapitel [Abschnitt 4.1.1 \[Allgemeine Vorschläge\]](#), Seite 44 gegeben, aber für große Projekte ist es unumgänglich. Muss z. B. die Definition für `fdannp` verändert werden, so braucht man es nur einmal vorzunehmen und die Noten in der Geigenstimme, `violin`, bleiben unberührt.

```
fdannp = _\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c' {
  g4\fdannp c'8. e16
}
```

4.2 Tipparbeit sparen durch Bezeichner und Funktionen

Bis jetzt haben Sie immer etwa solche Noten gesehen:

```
hornNotes = \relative c'' { c4 b dis c }
\score {
  {
    \hornNotes
  }
}
```



Das könnte auch nützlich in Minimal-Music sein:

```
fragA = \relative c'' { a4 a8. b16 }
fragB = \relative c'' { a8. gis16 ees4 }
violin = \new Staff { \fragA \fragA \fragB \fragA }
\score {
  {
    \violin
  }
}
```



Sie können diese Bezeichner oder Variablen aber auch für (eigene) Einstellungen verwenden:

```
dolce = \markup{ \italic\bold dolce }
padText = { \once \override TextScript #'padding = #5.0 }
fthenp=_markup{ \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
  \repeat volta 2 {
    c4._\dolce b8 a8 g a b |
    \padText
    c4.^"hi there!" d8 e' f g d |
    c,4.\fthenp b8 c4 c-. |
  }
}
\score {
  {
    \violin
  }
}
\layout{ragged-right=##t}
}
```



Die Variablen haben in diesem Beispiel deutlich die Tipparbeit erleichtert. Aber es lohnt sich, sie zu einzusetzen, auch wenn man sie nur einmal anwendet, denn Sie vereinfachen die Struktur. Hier ist das vorangegangene Beispiel ohne Variablen. Es ist sehr viel komplizierter zu lesen, besonders die letzte Zeile.

```
violin = \relative c'' {
  \repeat volta 2 {
    c4._\markup{ \italic \bold dolce } b8 a8 g a b |
    \once \override TextScript #'padding = #5.0
    c4.^"hi there!" d8 e' f g d |
    c,4.\markup{ \dynamic f \italic \small { 2nd }
      \hspace #0.1 \dynamic p } b8 c4 c-. |
  }
}
```

Bis jetzt wurde nur statische Substitution vorgestellt – wenn LilyPond den Befehl `\padText` findet, wird er ersetzt durch unsere vorherige Definition (alles, was nach dem `padtext=` kommt).

LilyPond kennt aber auch nicht-statische Substitutionen (man kann sie sich als Funktionen vorstellen).

```
padText =
#(define-music-function (parser location padding) (number?)
  #{
    \once \override TextScript #'padding = #$padding
  #})

\relative c''' {
  c4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" d e f
  \padText #2.6
  c4^"piu mosso" fis a g
}
```



Die Benutzung von Variablen hilft auch viele Schreibarbeit zu vermeiden, wenn die Eingabesyntax von LilyPond sich verändert (siehe auch [Abschnitt 4.4 \[Alte Dateien aktualisieren\]](#), Seite 51). Wenn nur eine einzige Definition (etwa `\dolce`) für alle Dateien verwendet wird (vgl. [Abschnitt 4.3 \[Stil-Dateien\]](#), Seite 48), muss nur diese einzige Definition verändert werden, wenn sich die Syntax ändert. Alle Verwendungen des Befehles beziehen sich dann auf die neue Definition.

4.3 Stil-Dateien

Die Ausgabe, die LilyPond erstellt, kann sehr stark modifiziert werden, siehe [Kapitel 5 \[Die Ausgabe verändern\]](#), [Seite 53](#) für Einzelheiten. Aber wie kann man diese Änderungen auf eine ganze Serie von Dateien anwenden? Oder die Einstellungen von den Noten trennen? Das Verfahren ist ziemlich einfach.

Hier ist ein Beispiel. Es ist nicht schlimm, wenn Sie nicht auf Anhieb die Abschnitte mit den ganzen `#()` verstehen. Das wird im [Kapitel Abschnitt 5.6 \[Fortgeschrittene Optimierungen mit Scheme\]](#), [Seite 60](#) erklärt.

```
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line(:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \tempoMark "Poco piu mosso"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}
```



Es treten einige Probleme mit überlappenden Symbolen auf. Sie werden beseitigt mit den Tricks aus dem [Kapitel Abschnitt 5.1 \[Verschieben von Objekten\]](#), [Seite 53](#). Aber auch die `mpdolce` und `tempoMark`-Definitionen können verbessert werden. Sie produzieren das Ergebnis, das gewünscht ist, aber es wäre schön, sie auch in anderen Stücken verwenden zu können. Man könnte sie natürlich einfach kopieren und in die anderen Dateien einfügen, aber das ist lästig. Die Definitionen verbleiben auch in der Notendatei und diese `#()` sehen nicht wirklich schön aus. Sie sollen in einer anderen Datei versteckt werden:

```
%% speichern in einer Datei "definitions.ly"
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line(:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})
```

Jetzt muss natürlich noch die Notendatei angepasst werden (gespeichert unter dem Namen "music.ly").

```
\include "definitions.ly"
```

```

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \once \override Score.RehearsalMark #'padding = #2.0
  \tempoMark "Poco piu mosso"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}

```



Das sieht schon besser, aber es sind noch einige Verbesserungen möglich. Das Glissando ist schwer zu sehen, also soll es etwas dicker erscheinen und dichter an den Notenköpfen gesetzt werden. Das Metronom-Zeichen soll über dem Schlüssel erscheinen, nicht über der ersten Note. Und schließlich kann unser Kompositionsprofessor „C“-Taktangaben überhaupt nicht leiden, also müssen sie in „4/4“ verändert werden.

Diese Veränderungen sollten Sie aber nicht in der ‘music.ly’-Datei vornehmen. Ersetzen Sie die ‘definitions.ly’-Datei hiermit:

```

%%% definitions.ly
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
  #{
    \once \override Score . RehearsalMark #'self-alignment-X = #left
    \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
    \mark \markup { \bold $markp }
  })

\layout{
  \context { \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context { \Staff
    \override TimeSignature #'style = #'numbered
  }
  \context { \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}

```



Das sieht schon besser aus! Aber angenommen ich möchte dieses Stück jetzt veröffentlichen. Mein Kompositionsprofessor mag die „C“-Taktangaben nicht, aber ich finde sie irgendwie schöner. Also kopiere ich die Datei ‘definitions.ly’ nach ‘web-publish.ly’ und verändere diese. Weil die Noten in einem Pdf auf dem Bildschirm angezeigt werden sollen, bietet es sich auch an, die gesamte Ausgabe zu vergrößern.

```
%% definitions.ly
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
  #{
    \once \override Score . RehearsalMark #'self-alignment-X = #left
    \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
    \mark \markup { \bold $markp }
  })

#(set-global-staff-size 23)
\layout{
  \context { \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context { \Staff
  }
  \context { \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



In der Notendatei muss jetzt nur noch `\include "definitions.ly"` durch `\include "web-publish.ly"` ausgetauscht werden. Das könnte man natürlich noch weiter vereinfachen. Also eine Datei ‘definitions.ly’, die nur die Definitionen von `mpdolce` und `tempoMark` enthält, eine Datei ‘web-publish.ly’, die alle die Änderungen für den `\layout`-Abschnitt enthält und eine Datei ‘university.ly’ für eine Ausgabe, die den Wünschen des Professors entspricht. Der Anfang der ‘music.ly’-Datei würde dann so aussehen:

```

\include "definitions.ly"

%%% Nur eine der beiden Zeilen auskommentieren!
\include "web-publish.ly"
%\include "university.ly"

```

Durch diese Herangehensweise kann auch bei der Erstellung von nur einer Ausgabeversion Arbeit gespart werden. Ich benutze ein halbes Dutzend verschiedener Stilvorlagen für meine Projekte. Jede Notationsdatei fängt an mit `\include "../global.ly"`, welches folgenden Inhalt hat:

```

%%% global.ly
\version "2.11.15"
#(ly:set-option 'point-and-click #f)
\include "../init/init-defs.ly"
\include "../init/init-layout.ly"
\include "../init/init-headers.ly"
\include "../init/init-paper.ly"

```

4.4 Alte Dateien aktualisieren

Die Syntax von LilyPond verändert sich ab und zu. Wenn LilyPond besser wird, muss auch die Syntax (Eingabesprache) entsprechend angepasst werden. Teilweise machen diese Veränderungen die Eingabesprache einfacher lesbar, teilweise dienen sie dazu, neue Eigenschaften des Programmes benutzbar zu machen.

LilyPond stellt ein Programm bereit, das Aktualisierungen vereinfacht: `convert-ly`. Einzelheiten zur Programmbenutzung finden sich in `program usage manual`, `Updating files with convert-ly`.

Leider kann `convert-ly` nicht alle Veränderungen der Syntax berücksichtigen. Hier werden einfache „Suchen und Ersetzen“-Veränderungen vorgenommen (wie etwa `raggedright` zu `becoming ragged-right`), aber einige Veränderungen sind zu kompliziert. Die Syntax-Veränderungen, die das Programm nicht berücksichtigt, sind im Kapitel `program usage manual`, `Updating files with convert-ly` aufgelistet.

Zum Beispiel wurden in LilyPond 2.4 und früheren Versionen Akzente und Umlaute mit LaTeX-Befehlen eingegeben, ein „No\"el“ etwa ergäbe das französische Wort für Weihnachten. In LilyPond 2.6 und höher müssen diese Sonderzeichen direkt als utf-8-Zeichen eingegeben werden, in diesem Fall also „ë“. `convert-ly` kann nicht alle dieser LaTeX-Befehle verändern, das muss manuell vorgenommen werden.

4.5 Fehlersuche (alles auseinander bauen)

Früher oder später werden Sie in die Lage kommen, dass LilyPond ihre Datei nicht kompilieren will. Die Information, die LilyPond während der Übersetzung gibt, können Ihnen helfen, den Fehler zu finden, aber in vielen Fällen müssen Sie nach der Fehlerquelle auf die Suche gehen.

Die besten Hilfsmittel sind in diesem Fall das Zeilen- und Blockkommentar (angezeigt durch `%` bzw. `{ ... }`). Wenn Sie nicht bestimmen können, wo sich das Problem befindet, beginnen Sie damit, große Teile des Quelltextes auszukommentieren. Nachdem Sie einen Teil auskommentiert haben, versuchen Sie, die Datei erneut zu übersetzen. Wenn es jetzt funktioniert, muss sich das Problem innerhalb der Kommentare befinden. Wenn es nicht funktioniert, müssen Sie weitere Teile auskommentieren bis sie eine Version haben, die funktioniert.

In Extremfällen bleibt nur noch solch ein Beispiel übrig,

```

\score {
  <<

```

```

% \melody
% \harmony
% \bass
>>
\layout{}
}

```

(also eine Datei ohne Noten).

Geben Sie nicht auf, wenn das vorkommen sollte. Nehmen Sie das Kommentarzeichen on einem Teil wieder weg, sagen wir der Bassstimme und schauen Sie, ob es funktioniert. Wenn nicht, dann kommentieren sie die gesamte Bassstimme aus, aber nicht den `\bass`-Befehl in dem `\score`-Abschnitt:

```

bass = \relative c' {
%{
  c4 c c c
  d d d d
%}
}

```

Jetzt beginnen Sie damit, langsam Stück für Stück der Bassstimme wieder hineinzunehmen, bis Sie die problematische Zeile finden.

Eine andere nützliche Technik zur Problemlösung ist es, [Abschnitt 4.6 \[Minimalbeispiele\]](#), [Seite 52](#) zu konstruieren.

4.6 Minimalbeispiele

Ein Minimalbeispiel ist eine Beispieldatei, die so klein wie möglich ist. Diese Beispiele sind sehr viel einfacher zu verstehen als die langen Originaldateien. Minimalbeispiele werden benutzt, um

- Fehlerberichte zu erstellen,
- eine Hilfeanfrage an die E-Mail-Liste zu schicken,
- Ein Beispiel zur [LilyPond Schnipselsammlung](#) hinzuzufügen.

Um ein Beispiel zu konstruieren, dass so klein wie möglich ist, gibt es eine einfache Regel: Alles nicht Notwendige entfernen. Wenn Sie unnötige Teile einer Datei entfernen, bietet es sich an, sie auszukommentieren und nicht gleich zu löschen. Auf diese Weise können Sie eine Zeile leicht wieder mit aufnehmen, sollten Sie sie doch brauchen, anstatt sie von Anfang an neu zu schreiben.

Es gibt zwei Ausnahmen dieser „So klein wie möglich“-Regel:

- Fügen Sie immer einen `\version`-Befehl ein.
- Wenn es möglich ist, benutzen Sie `\paper{ ragged-right=##t }` am Beginn des Beispiels.

Der Sinn der Minimalbeispiel ist, dass sie einfach lesbar sind:

- Vermeiden Sie es, komplizierte Noten, Schlüssel oder Taktangaben zu verwenden, es sei denn, Sie wollen genau an diesen Elementen etwas demonstrieren.
- Benutzen Sie keine `\override`-Befehle, wenn Sie nicht der Zweck des Beispiels sind.

5 Die Ausgabe verändern

In diesem Kapitel wird erklärt, wie man die Notenausgabe verändern kann. In LilyPond kann man sehr viel konfigurieren, fast jedes Notenfragment kann geändert werden.

5.1 Verschieben von Objekten

Es wird vielleicht eine Überraschung sein, aber LilyPond ist nicht perfekt. Einige Notationselemente können sich überschneiden. Das ist nicht schön, kann aber (in den meisten Fällen) sehr einfach korrigiert werden.

TODO: Mit den neuen Abstandseigenschaften seit Version 2.12 sind die jeweiligen Beispiele nicht mehr relevant. Sie zeigen jedoch immer noch machtvolle Eigenschaften von LilyPond und verbleiben deshalb in der Dokumentation, bis jemand bessere Beispiel zur Verfügung stellt.

```
% temporary code to break this example:
\override TextScript #'outside-staff-priority = ##f
e4^\markup{ \italic ritenuto } g b e
```



Die einfachste Lösung ist es, Abstände zwischen Objekt und Note zu vergrößern (genauso auch für Fingersätze oder Dynamikzeichen). In LilyPond wird das durch Veränderung der `padding` (Füllungs)-Eigenschaft erreicht, ihre Maßeinheit sind Notenzeilenabstände. Für die meisten Objekte ist der Wert etwa 1.0 oder weniger (das unterscheidet sich von Objekt zu Objekt). Hier soll der Abstand vergrößert werden, also scheint 1.5 eine gute Wahl.

```
% temporary code to break this example:
\override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'padding = #1.5
e4^\markup{ \italic ritenuto } g b e
```



Das sieht besser aus, ist aber noch nicht groß genug. Nach einigen Experimenten wird darum 2.3 genommen für diesen Fall. Diese Zahl ist aber nur das Resultat einigen Probierens und persönlicher Geschmack. Probieren Sie selber ein wenig herum und entscheiden Sie nach eigenem Geschmack.

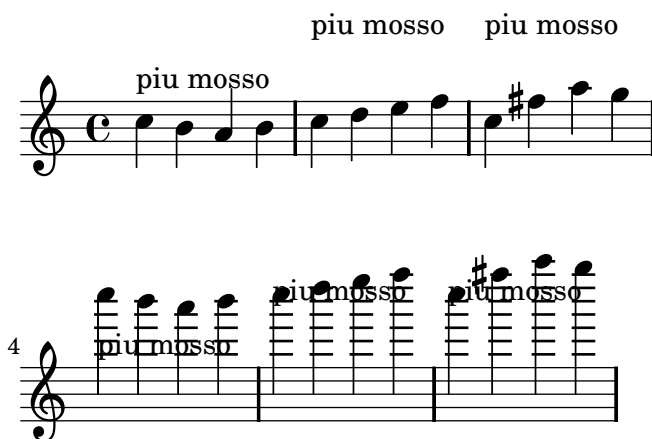
Die `staff-padding`-Eigenschaft ist der vorigen sehr ähnlich. `padding` entscheidet über den minimalen Abstand zwischen einem Objekt und dem nächsten anderen Objekt (meistens eine Note oder Notenzeile); `staff-padding` entscheidet über den minimalen Abstand zwischen einem Objekt und dem Notensystem. Das ist nur ein kleiner Unterschied, aber hier wird das Verhalten demonstriert:

```
% temporary code to break this example:
\override TextScript #'outside-staff-priority = ##f
c4~"piu mosso" b a b
\once \override TextScript #'padding = #4.6
c4~"piu mosso" d e f
```

```

\once \override TextScript #'staff-padding = #4.6
c4^"piu mosso" fis a g
\break
c'4^"piu mosso" b a b
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
\once \override TextScript #'staff-padding = #4.6
c4^"piu mosso" fis a g

```



Eine andere Lösung ermöglicht vollständige Kontrolle über die Positionierung eines Objektes sowohl horizontal als auch vertikal. Das wird mit der `extra-offset` (Zusätzlicher-Abstand)-Eigenschaft erreicht. Das ist etwas komplizierter und kann andere Probleme mit sich ziehen. Wenn Objekte mit dieser Eigenschaft verschoben werden, heißt das, dass LilyPond sie erst setzt, nachdem alle anderen Objekte positioniert worden sind. Deshalb können sich die Objekte am Ende überlagern.

```

% temporary code to break this example:
\override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'extra-offset = #'( 1.0 . -1.0 )
e4^\markup{ \italic ritenuto } g b e

```



Bei Verwendung von `extra-offset` bestimmt die erste Zahl über die horizontale Verschiebung (nach links ist negativ), die zweite Zahl bestimmt die vertikale Verschiebung (nach oben ist positiv). Nach einigen Experimenten wurden hier folgende Werte für gut befunden:

```

% temporary code to break this example:
\override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'extra-offset = #'( -1.6 . 1.0 )
e4^\markup{ \italic ritenuto } g b e

```



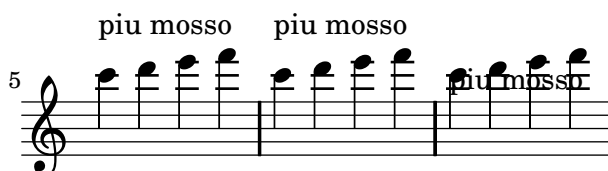
Auch diese Zahlen sind nur Resultat einigen Herumprobierens und Vergleichens der Ergebnisse. Sie wollen den Text vielleicht etwas höher oder etwas mehr nach links setzen. Versuchen Sie es selber und vergleichen Sie das Ergebnis.

Eine letzte Warnung: in diesem Kapitel haben wir den Befehl

```
\once \override TextScript ...
```

benutzt. Dieser Befehl verändert die Anzeige des Textes für die nächste Note. Wenn die Note keinen Text zugeordnet hat, wird auch nichts verändert (und es wird **nicht** nach dem nächsten Text gesucht). Um das Verhalten zu verändern, so dass alles, was nach dem Befehl kommt, verändert wird, müssen Sie den Befehl `\once` weglassen. Um die Veränderung zu stoppen, benutzen Sie den Befehl `\revert`. Das wird genauer im Kapitel [Abschnitt 9.3 \[The \override command\]](#), Seite 134 erklärt.

```
% temporary code to break this example:
\override TextScript #'outside-staff-priority = ##f
c4^"piu mosso" b
\once \override TextScript #'padding = #4.6
a4 b
c4^"piu mosso" d e f
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
c4^"piu mosso" d e f
\break
\override TextScript #'padding = #4.6
c4^"piu mosso" d e f
c4^"piu mosso" d e f
\revert TextScript #'padding
c4^"piu mosso" d e f
```



Siehe auch

[Abschnitt 9.3 \[The \override command\]](#), Seite 134, [Abschnitt 5.3 \[Übliche Optimierungen\]](#), Seite 56.

5.2 Überlappende Notation in Ordnung bringen

Im Kapitel [Abschnitt 5.1 \[Verschieben von Objekten\]](#), Seite 53 wurde gezeigt, wie man Texte (`TextScript`-Objekte) verschiebt. Mit der gleichen Technik können auch andere Objektklassen verschoben werden, `TextScript` muss dann nur durch den Namen des Objektes ersetzt werden.

Um den Objektnamen zu finden, siehe die ‚**see also**‘-Hinweise am Ende des jeweiligen Abschnittes. Zum Beispiel am Ende des Kapitels [Abschnitt 6.6.3 \[Dynamik\]](#), Seite 104 findet sich:

Siehe auch

Programmreferenz: `DynamicText`, `Hairpin`. Vertikale Positionierung dieser Symbole wird mit `DynamicLineSpanner` erreicht.

Um also Dynamik-Zeichen zu verschieben, muss

```
\override DynamicLineSpanner #'padding = #2.0
```

benutzt werden. Es ist nicht genügend Platz, um jedes Objekt aufzulisten, aber die gebräuchlichsten finden sich hier:

Objekttyp	Objektbezeichnung
Dynamikzeichen (vertikal)	<code>DynamicLineSpanner</code>
Dynamikzeichen (horizontal)	<code>DynamicText</code>
Bindebögen	<code>Tie</code>
Phrasierungsbögen	<code>Slur</code>
Artikulationszeichen	<code>Script</code>
Fingersatz	<code>Fingering</code>
Text, z. B. <code>^"text"</code>	<code>TextScript</code>
Übungs-/Textmarken	<code>RehearsalMark</code>

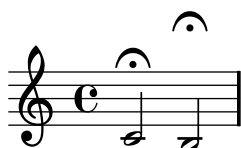
5.3 Übliche Optimierungen

Bestimmte Korrekturen sind so häufig, dass für sie schon fertige angepasste Befehle bereitgestellt sind, so etwa `\slurUp` um einen Bindebogen oberhalb anzuzeigen oder `\stemDown` um den Notenhals nach unten zu zwingen. Diese Befehle sind im Teil Alles über die Notation unter dem entsprechenden Abschnitt erklärt.

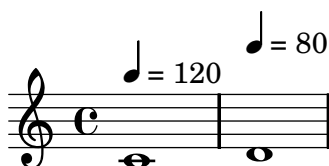
Eine vollständige Liste aller Veränderungen, die für jeden Objekttypen (etwa Bögen oder Balken) zur Verfügung stehen, ist in der Programmreferenz dargestellt. Viele Layoutobjekte benutzen jedoch gleiche Eigenschaften, die benutzt werden können, um eigene Einstellungen vorzunehmen.

- `padding`-Eigenschaft kann gesetzt werden, um den Abstand zwischen Symbolen über oder unter den Noten zu vergrößern oder zu verkleinern. Das gilt für alle Objekte, die ein `side-position-interface` besitzen, also unterscheiden, auf welcher Seite der Note sie sich befinden.

```
c2\fermata
\override Script #'padding = #3
b2\fermata
```



```
% This will not work, see below:
\override MetronomeMark #'padding = #3
\tempo 4=120
c1
% This works:
\override Score.MetronomeMark #'padding = #3
\tempo 4=80
d1
```



Im zweiten Beispiel ist es sehr wichtig zu wissen, welcher Kontext für bestimmte Objekte zuständig ist. Weil das `MetronomeMark`-Objekt vom `Score`-Kontext gesetzt wird, werden Veränderungen innerhalb des `Voice`-Kontextes nicht berücksichtigt. Genauere Details im Kapitel [Abschnitt 9.3.1 \[Eine Korrektur konstruieren\]](#), Seite 134.

- Die `extra-offset`-Eigenschaft verschiebt Objekte, hier ist ein Zahlenpaar zur Angabe der Positionierung erforderlich. Die erste Nummer bestimmt die horizontale Bewegung, eine positive Zahl bewegt das Objekt nach rechts. Die zweite Zahl bestimmt die vertikale Bewegung, eine positive Zahl bewegt das Objekt nach oben. Die `extra-offset`-Eigenschaft läuft auf unterster Ebene ab: Die Formatierungsmaschine ist sich der Veränderungen nicht bewusst.

Im folgenden Beispiel wird die zweite Fingersatzbezeichnung etwas nach links verschoben und 1,8 Notenzeilenabstände nach unten:

```
\stemUp
f-5
\once \override Fingering
  #'extra-offset = #'(-0.3 . -1.8)
f-5
```



- Die Verwendung der `transparent`-Eigenschaft druckt das entsprechende Objekt mit „unsichtbarer Druckerschwärze“: Das Objekt wird nicht angezeigt, aber sein Verhalten bleibt bestehen. Das Objekt nimmt weiterhin Platz ein, es nimmt teil an Überschneidungen und deren Auflösung durch das Programm, Bögen und Balken können daran angebunden werden.

Das nächste Beispiel zeigt, wie man unterschiedliche Stimmen mit Bindebögen verbinden kann. Normalerweise können Bindebögen nur zwei Noten der selben Stimme verbinden. Indem aber ein Bogen in einer anderen Stimme erstellt wird,



und dann der erste Hals nach oben unsichtbar gemacht wird, scheint der Bindebogen die Stimme zu wechseln:

```
<< {
  \once \override Stem #'transparent = ##t
  b8~ b8\noBeam
} \ {
  b[ g8]
} >>
```



Damit der Hals den Bogen nicht zu sehr verkleinert, wird seine Länge (`length`) auf den Wert 8 gesetzt:

```
<< {
  \once \override Stem #'transparent = ##t
  \once \override Stem #'length = #8
  b8~ b8\noBeam
} \ {
  b[ g8]
} >>
```



Abstände in LilyPond werden in Notenzeilenabständen (`staff-space`) gemessen, während die meisten Dicke-Eigenschaften auf mit der Notenliniendicke korrespondieren. Eine Eigenschaft verhalten sich anders, etwa die Dicke von Balken ist an die Notenzeilenabstände gekoppelt. Mehr Information findet sich im relevanten Teil der Programmreferenz.

5.4 Standarddateien

Die Programmreferenz enthält sehr viel Information über LilyPond, aber noch mehr Information findet sich in den internen LilyPond-Dateien.

Eine Standardeinstellungen (wie die Definitionen für den Kopf (`\header`) sind als `.ly`-Datei gespeichert. Andere Einstellungen (wie die Definition für Beschriftung (`markup`) sind als `.scm` (Scheme)-Datei gespeichert. Eine nähere Erklärung geht über den Rahmen dieses Handbuchs hinaus. Der Hinweis scheint aber angebracht, dass es grundlegende technische Kenntnis und sehr viel Zeit erfordert, diese Dateien zu verstehen.

- Linux: `'installdir/lilypond/usr/share/lilypond/current/'`
- OS X: `'installdir/LilyPond.app/Contents/Resources/share/lilypond/current/'`.
Um diese Ordner anzuschauen, wechseln Sie entweder mit `cd` im Terminal zu der Adresse oder klicken Sie mit der rechten Maustaste auf das LilyPond-Symbol und wählen Sie ‚Show Package Contents‘.
- Windows: `'installdir/LilyPond/usr/share/lilypond/current/'`

Die `'ly/'` und `'scm/'`-Ordner sind von besonderem Interesse. Dateien wie `'ly/property-init.ly'` und `'ly/declarations-init.ly'` definieren alle häufig vorkommenden Veränderungen.

5.5 Die Musik auf weniger Seiten zwingen

Manchmal bleiben nur noch ein oder zwei Systeme auf der letzten Seite übrig. Das ist immer ärgerlich, besonders wenn es scheint, dass auf den vorigen Seiten genug Platz ist, um die Systeme noch unterzubringen.

Wenn man versucht, das Layout zu verändern, kommt einem der Befehl `annotate-spacing` zu Hilfe. Mit diesem Befehl erhält man die Werte von verschiedenen Abstandsbefehlen ausgedruckt, mehr Information im Kapitel [Abschnitt 11.3 \[Abstände anzeigen lassen\]](#), Seite 138. Anhand dieser Angaben kann dann entschieden werden, welche Werte verändert werden müssen.

Neben Rändern gibt es nämlich weitere Optionen, Platz zu sparen:

- LilyPond kann die Systeme so dicht wie möglich platzieren (damit so viele Systeme wie möglich auf eine Seite passen), aber sie dann so anordnen, dass kein weißer Rand unten auf der Seite entsteht.

```
\paper {
  between-system-padding = #0.1
  between-system-space = #0.1
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

- Die Anzahl der Systeme kann reduziert werden (wenn LilyPond die Musik auf 11 Systeme verteilt, kann man die Benutzung von nur 10 Systemen erzwingen).

```
\paper {
  system-count = #10
}
```

- Vermeidung von Objekten, die den vertikalen Abstand von Systemen vergrößern, hilft oft. Die Verwendung von Klammern bei Wiederholungen etwa braucht mehr Platz. Wenn die Noten innerhalb der Klammern auf zwei Systeme verteilt sind, brauchen sie mehr Platz, als wenn sie nur auf einer Zeile gedruckt werden.

Ein anderes Beispiel ist es, Dynamik-Zeichen, die besonders weit „hervorstehen“, zu verschieben.

```
\relative c' {
  e4 c g\ff c
  \override DynamicLineSpanner #'padding = #-1.8
  \override DynamicText #'extra-offset = #'(-2.1 . 0)
  e4 c g\ff c
}
```



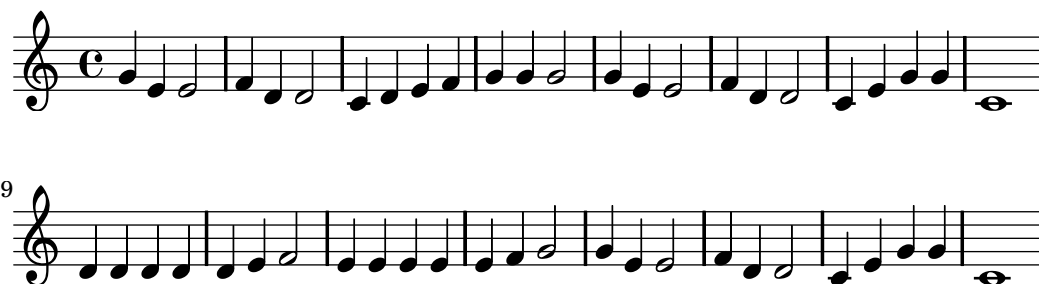
- Die horizontalen Abstände können mit der `SpacingSpanner`-Eigenschaft verändert werden. Siehe [Abschnitt 11.6.3 \[Horizontale Abstände verändern\]](#), Seite 139 für Einzelheiten.

```
\score {
  \relative c'' {
    g4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
  }
}
```

```

      g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    }
  \layout {
    \context {
      \Score
      \override SpacingSpanner
        #'base-shortest-duration = #(ly:make-moment 1 4)
    }
  }
}

```



5.6 Fortgeschrittene Optimierungen mit Scheme

Es wurde schon gezeigt, wie die LilyPond-Ausgabe sehr stark verändert werden kann, indem man Befehle wie `\override TextScript #'extra-offset = (1 . -1)` benutzt. Aber noch mehr Einfluss auf die Formatierung kann durch den Einsatz von Scheme genommen werden. Eine vollständige Erklärung findet sich in der [Anhang B \[Scheme-Übung\], Seite 146](#) und den [Kapitel 12 \[Schnittstellen für Programmierer\], Seite 140](#).

Scheme kann benutzt werden, um einfach nur Befehle zu „überschreiben“ (`\override`):

```

padText = #(define-music-function (parser location padding) (number?)
  #{
    \once \override TextScript #'padding = #$padding
  #})

\relative c''' {
  c4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" d e f
  \padText #2.6
  c4^"piu mosso" fis a g
}

```



Hiermit können aber auch neue Befehle erstellt werden:

```

tempoMark = #(define-music-function (parser location padding marktext)
  (number? string?)
  #{

```

```

\once \override Score . RehearsalMark #'padding = $padding
\once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
\mark \markup { \bold $marktext }
#})

\relative c'' {
  c2 e
  \tempoMark #3.0 #"Allegro"
  g c
}

```

Allegro



Sogar ganze musikalische Ausdrücke können eingefügt werden:

```

pattern = #(define-music-function (parser location x y) (ly:music? ly:music?)
#{
  $x e8 a b $y b a e
#})

\relative c''{
  \pattern c8 c8\f
  \pattern {d16 dis} { ais16-> b\p }
}

```



5.7 Vermeiden von Optimierungen durch langsamere Übersetzung

LilyPond kann einige zusätzliche Tests durchführen, während die Noten gesetzt werden. Dadurch braucht das Programm länger, um den Notensatz zu produzieren, aber üblicherweise werden weniger nachträgliche Anpassungen nötig sein.

```

%% Um sicher zu gehen, dass Texte und Liedtext
%% innerhalb der Papierränder bleiben
\override Score.PaperColumn #'keep-inside-line = ##t

```

6 Grundlegende Notation

In diesem Kapitel wird die Notation der am häufigsten benutzten Notationsformen und -elemente vorgestellt.

6.1 Tonhöhen

In diesem Abschnitt wird behandelt, wie die Tonhöhe angegeben wird.

6.1.1 Normale Tonhöhen

Tonhöhenbezeichnungen werden durch Kleinbuchstaben von **a** bis **g** angegeben.¹ Eine aufsteigende C-Dur-Tonleiter wird wie folgt notiert:

```
\clef bass
c d e f g a b c'
```



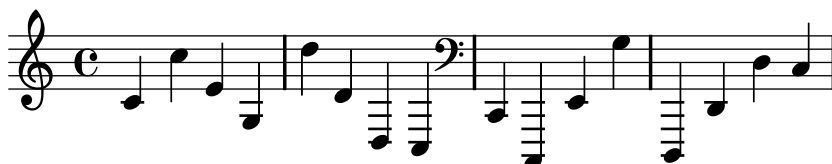
Die Notenbezeichnung **c** schreibt als Note ein kleines C, eine Oktave unter dem eingestrichenen C.

```
\clef treble
c1
\clef bass
c1
```



Zusätzliche Oktavbestimmung wird mit einer Anzahl von Apostrophen (',') oder Kommas (,,) vorgenommen. Jeder Apostroph erhöht die Note um eine Oktave, jedes Komma erniedrigt sie um eine Oktave.

```
\clef treble
c' c'' e' g d'' d' d c
\clef bass
c, c,, e, g d,, d, d c
```



Eine alternative Methode gibt am Anfang die Oktave vor, innerhalb derer die Noten gesetzt werden, dabei werden unter Umständen weniger Oktavangaben (' oder ,) benötigt. Siehe auch [Abschnitt 6.1.6 \[Relative Oktaven\], Seite 65](#).

¹ Die Benutzung deutscher Notenbezeichnungen mit der Unterscheidung von **b** und **h** ist auch möglich, siehe [Abschnitt 6.1.5 \[Notenbezeichnungen in anderen Sprachen\], Seite 65](#).

6.1.2 Versetzungszeichen

Ein Kreuz wird eingegeben, indem man `-is` an die Notenbezeichnung hängt, ein `b` durch `-es`. Doppelkreuze und Doppel-Bs werden durch Hinzufügen von `-isis` und `-eses` hinter die Notenbezeichnung erzeugt.

```
a2 ais a aes
a2 aisis a aeses
```



Auch die deutschen Varianten `as` für `aes` und `es` für `ees` sind erlaubt. Im Unterschied zum Deutschen ist aber `bes` die einzige Version für den Ton B, während `his` als `bis` geschrieben werden muss. Das kann aber auch verändert werden, siehe [Abschnitt 6.1.5 \[Notenbezeichnungen in anderen Sprachen\]](#), Seite 65.

```
a2 as e es
```



Ein Auflösungszeichen macht die Wirkung eines Kreuzes oder Bs rückgängig. Diese Auflösungszeichen werden jedoch nicht als Suffix einer Tonhöhenbezeichnung eingegeben, sondern sie ergeben sich (automatisch) aus dem Kontext, wenn die nicht alterierte Notenbezeichnung eingegeben wird.

```
a4 aes a2
```



Die Sequenz `d e f` wird interpretiert als: „Setze eine D-Noten, eine E-Note und eine F-Note,“ unabhängig von den Vorzeichen der Tonart. Mehr Information über den Unterschied zwischen musikalischem Inhalt und der Präsentation dieses Inhalts siehe [Abschnitt 2.2.2 \[Versetzungszeichen und Tonartbezeichnung \(Vorzeichen\)\]](#), Seite 18.

```
\key d \major
d e f g
d e fis g
```



Übliche Veränderungen der Einstellungen

Den Satzregeln für den Notensatz folgend wird ein Auflösungszeichen dann ausgegeben, wenn eine vorhergehende Akzidenz rückgängig gemacht werden soll. Um dieses Verhalten zu ändern, muss `\set Staff.extraNatural = ##f` eingesetzt werden.

```
ceses4 ces cis c
\set Staff.extraNatural = ##f
ceses4 ces cis c
```



Siehe auch

Programmreferenz: `LedgerLineSpanner`, `NoteHead`.

6.1.3 Warnungsversetzungszeichen

Normalerweise werden Versetzungszeichen automatisch gesetzt, aber sie können auch manuell hinzugefügt werden. Ein erinnerndes Versetzungszeichen kann erzwungen werden, indem man ein Ausrufungszeichen (!) hinter die Notenbezeichnung schreibt. Ein warnendes Versetzungszeichen (also ein Vorzeichen in Klammern) wird durch Anfügen eines Fragezeichens (?) erstellt. Mit diesen zusätzlichen Zeichen kann man sich auch Auflösungszeichen ausgeben lassen.

```
cis cis cis! cis? c c? c! c
```



Siehe auch

Die automatische Setzung von Versetzungszeichen kann auf viele Arten beeinflusst werden. Mehr Information dazu siehe [Abschnitt 9.1.1 \[Automatische Versetzungszeichen\]](#), Seite 134.

6.1.4 Mikrotöne

Versetzungszeichen für Vierteltöne werden durch Anhängen der Endungen `-eh` (Erniedrigung) und `-ih` (Erhöhung) an den Tonhöhenbuchstaben erstellt. Das Beispiel zeigt eine in Vierteltönen aufsteigende Serie vom kleinen C.

```
\set Staff.extraNatural = ##f
ceseh ceh cih cisih
```



Mikrotöne werden auch in die MIDI-Dateien geschrieben.

Fehler

Es gibt keine allgemein anerkannten Standards für die Notation von Dreiviertelton-Erniedrigungszeichen. LilyPonds Symbol entspricht also keinem Standard.

6.1.5 Notenbezeichnungen in anderen Sprachen

Es gibt vordefinierte Bezeichnungen für die Notenbezeichnungen in anderen Sprachen als Englisch. Um sie zu benutzen, muss nur die entsprechende Datei für die jeweilige Sprache eingefügt werden. Zum Beispiel fügt man mit `\include "deutsch.ly"` die Notendefinitionen für die deutsche Sprache am Anfang der Datei hinzu. In der Tabelle sind die existierenden Definitionen mit den dazugehörigen Notenbezeichnungen dargestellt.

	Notenbezeichnungen								Kreuz	B	Doppelkreuz	Dopp
nederlands.ly	c	d	e	f	g	a	bes	b	-is	-es	-isis	-es
english.ly	c	d	e	f	g	a	bf	b	-s/-sharp	-f/-flat	-ss/-x/ -sharpsharp	-ff/ -fla
deutsch.ly	c	d	e	f	g	a	b	h	-is	-es	-isis	-es
norsk.ly	c	d	e	f	g	a	b	h	-iss/-is	-ess/-es	-ississ/-isis	-ess
svenska.ly	c	d	e	f	g	a	b	h	-iss	-ess	-ississ	-ess
italiano.ly	do	re	mi	fa	sol	la	sib	si	-d	-b	-dd	-bb
catalan.ly	do	re	mi	fa	sol	la	sib	si	-d/-s	-b	-dd/-ss	-bb
espanol.ly	do	re	mi	fa	sol	la	sib	si	-s	-b	-ss	-bb

Auf Holländisch, Deutsch, Norwegisch und Schwedisch (u. a.) werden die Erniedrigungen von ‚a‘ wie `aes` und `aeses` zu `as` und `ases` (oder auch `asas`) zusammengezogen. In manchen Sprachen sind nur diese Kurzformen definiert (das gilt auch für die Endungen der Vierteltöne).

Bestimmte Musik verwendet Alterationen, die Bruchteile von den „üblichen“ Kreuzen oder Bs sind. Die Notenbezeichnungen für Vierteltöne für die verschiedenen Sprachen sind in der folgenden Tabelle aufgeführt. Die Präfixe „Semi-“ und „Sesqui-“ bedeuten „halb“ bzw. „eineinhalb“. Für Norwegisch, Schwedisch, Katalanisch und Spanisch sind noch keine eigenen Namen definiert.

	Notenbezeichnungen								Semi- kreuz	Semi- B	Sesqui- Kreuz	Sesqui- B
nederlands.ly	c	d	e	f	g	a	bes	b	-ih	-eh	-isih	-eseh
english.ly	c	d	e	f	g	a	bf	b	-qs	-qf	-tqs	-tqf
deutsch.ly	c	d	e	f	g	a	b	h	-ih	-eh	-isih	-eseh
norsk.ly	c	d	e	f	g	a	b	h				
svenska.ly	c	d	e	f	g	a	b	h				
italiano.ly	do	re	mi	fa	sol	la	sib	si	-sd	-sb	-dsd	-bsb
catalan.ly	do	re	mi	fa	sol	la	sib	si				
espanol.ly	do	re	mi	fa	sol	la	sib	si				

6.1.6 Relative Oktaven

Oktaven werden angegeben, indem man ‚‘ oder ‚,‘ an die Notenbezeichnung hängt. Wenn Sie schon existierende Musik kopieren, passiert es schnell, eine Note aus Versehen in die falsche Oktave zu setzen, und der Fehler ist schwer zu finden. Der relative Oktaven-Modus verhindert solche Fehler, indem mögliche Fehler stark vergrößert werden: ein einziger Oktavierungsfehler wirkt sich auf den gesamten Rest des Stückes aus.

Die Syntax des Befehls lautet:

```
\relative Referenzoktave musikalischer Ausdruck
```

oder:

`\relative` *musikalischer Ausdruck*

Das eingestrichene C (c') wird als Referenzoktave angenommen, wenn sie nicht extra angegeben wird.

Die Oktave von Noten, die im musikalischen Ausdruck notiert sind, wird wie folgt erschlossen: Wenn keine Oktavversetzungszeichen benutzt werden, wird als Intervall zwischen der Noten und der vorhergehenden immer eine Quarte oder kleiner angenommen. Dieser Abstand wird ohne Rücksicht auf Alterationen bestimmt. Eine übermäßige Quarte ist also ein kleineres Intervall als eine verminderte Quinte, auch wenn beide sechs Halbtöne groß sind.

Die Oktavversetzungszeichen ' und , können hinzugefügt werden, um die Tonhöhe um eine Oktave zu erhöhen oder zu erniedrigen. Wenn der relative Modus beginnt, kann ein Referenzton angegeben werden, der als die vorhergehende Note für die erste Tonhöhe des musikalischen Ausdrucks verwendet wird. Wenn dieser Referenzton nicht angegeben wird, wird das eingestrichene C verwendet.

So funktioniert der relative Modus:

```
\relative c'' {
  b c d c b c bes a
}
```



Oktavversetzungen müssen für alle Intervalle angezeigt werden, die größer als eine Quarte sind.

```
\relative c'' {
  c g c f, c' a, e''
}
```



Wenn der vorherige Ausdruck ein Akkord ist, wird die erste Note des Akkordes benutzt, um die erste Note des nächsten Akkordes zu bestimmen.

```
\relative c' {
  c <c e g>
  <c' e g>
  <c, e' g>
}
```



Die Tonhöhe nach `\relative` muss eine Notenbezeichnung enthalten.

Die relative Veränderung wirkt sich nicht auf Transposition (`\transpose`), Akkordnotation (`\chordmode`) oder `\relative`-Abschnitte aus. Um den relativen Modus innerhalb von transponierter Musik zu verwenden, muss ein zusätzliches `\relative` innerhalb der Klammern des `\transpose`-Befehls gesetzt werden.

6.1.7 Oktavenüberprüfung

Durch Oktavenüberprüfung können Fehler einfacher entdeckt werden: nach einer Note kann =Apostrophe geschrieben werden, womit angezeigt wird, was ihre wirkliche Oktave sein soll. Im folgenden Beispiel

```
\relative c'' { c='' b=' d,='' }
```

erzeugt das `d` eine Warnung, weil ein `d''` erwartet wird (denn zwischen `b'` und `d''` befindet sich nur eine Terz), aber ein `d'` ist notiert. In der Notenausgabe wird die Oktave zu `d''` korrigiert und die nächste Note wird relativ zu `d''` anstelle von `d'` errechnet.

Es gibt auch eine Oktavenüberprüfung, die keine sichtbare Ausgabe erzeugt. Die Syntax:

```
\octave Tonhöhe
```

Hierdurch wird überprüft, dass die *Tonhöhe* (ohne Apostroph) der *Tonhöhe* (mit Apostroph) entspricht. Wenn sie sich nicht entsprechen, wird eine Warnung ausgegeben und die Oktave wird korrigiert. Die *Tonhöhe* wird nicht als Note gesetzt.

Im nächsten Beispiel erzeugt die erste Überprüfung keine Warnung, weil das `e` (im relativen Modus) innerhalb einer Quarte zum `a'` liegt. Die zweite Überprüfung aber erzeugt eine Warnung, weil das `e` mehr als eine Quarte vom `b'` entfernt ist. Die Warnung wird ausgegeben und die Oktave wird korrigiert, so dass auch die folgenden Noten wieder in der richtigen Oktave gesetzt werden.

```
\relative c' {
  e
  \octave a'
  \octave b'
}
```

Die Oktave einer Note, der eine Oktavüberprüfung angefügt wurde, wird in Hinsicht auf die vorherige Note bestimmt. Im nächsten Fragment ist die letzte Note ein `a'`. Die Oktavenüberprüfung stellt fest, dass zwischen `e'` und kleinem `b` eine Quarte Abstand ist und dass die folgende Note, ein `a'`, sich wieder innerhalb einer Quarte vom `e'` aus befindet. Die Überprüfung gibt also einen Erfolgswert zurück und die Notenausgabe wird nicht verändert.

```
\relative c' {
  e
  \octave b
  a
}
```



6.1.8 Transposition

Ein musikalischer Ausdruck kann mit dem Befehl `\transpose` transponiert werden. Die Syntax lautet:

```
\transpose von nach mus. Ausdruck
```

Das bedeutet, dass der *mus. Ausdruck* um das Intervall zwischen den Tonhöhen *von* und *nach* transponiert wird: Jede Note, die die Tonhöhe *von* hat, wird in die Tonhöhe *nach* umgewandelt.

So kann z. B. ein Stück in D-Dur, wenn es für den Sänger etwas zu tief ist, mit dem Befehl

```
\transpose d e ...
```

nach E-Dur transponiert werden.

Oder eine Violinstimme, die so notiert wird, wie sie erklingt, soll von einer A-Klarinette gespielt werden. Hier ist ein klingendes A als C notiert, so dass alles also eine kleine Terz tiefer erklingt, als es notiert ist. Für die Erzeugung der Klarinettenstimme muss folgender Befehl verwendet werden:

```
\transpose a c ...
```

`\transpose` unterscheidet enharmonische Verwechslungen: sowohl `\transpose c cis` als auch `\transpose c des` transponieren die Musik einen Halbton nach oben. Aber die erste Version gibt als Versetzungszeichen Kreuze aus, die zweite dagegen B-Versetzungszeichen.

```
mus = { \key d \major cis d fis g }
\new Staff {
  \clef "F" \mus
  \clef "G"
  \transpose c g' \mus
  \transpose c f' \mus
}
```



`\transpose` kann auch benutzt werden, um die geschriebenen Noten eines transponierenden Instruments zu notieren. Tonhöhen in LilyPond werden üblicherweise notiert, wie sie erklingen, aber man kann auch eine andere Tonart verwenden. Noten einer B-Trompete, die mit einem klingenden D anfangen, könnte man also auch so eingeben:

```
\transpose c bes { e4 ... }
```

Um die Noten dann wiederum als Trompetenstimme zu drucken (also einen Ganzton tiefer, als sie erklingen), setzt man einfach um sie herum eine weitere Transposition:

```
\transpose bes c { \transpose c bes { e4 ... } }
```

Siehe auch

Programmreferenz: `TransposedMusic`.

Beispiel: `'input/test/smart-transpose.ly'`.

Fehler

Wenn Sie sowohl `\transpose` als auch `\relative` benutzen wollen, muss die `\transpose`-Umgebung sich außerhalb der `\relative`-Umgebung befinden, da `\relative` keine Auswirkungen auf Noten hat, die sich innerhalb von `\transpose` befinden.

6.1.9 Pausen

Pausen werden wie Noten eingegeben, ihre Bezeichnung ist `r`.

```
r1 r2 r4 r8
```



Pausen, die ganze Takte ausfüllen und in der Taktmitte zentriert werden sollen, müssen als mehrtaktige Pausen eingegeben werden. Sie können sowohl für einen einzigen Takt als auch für mehrere Takte verwendet werden, Näheres im Abschnitt [Abschnitt 8.2.1 \[Mehraktige Pausen\]](#), [Seite 131](#).

Um die vertikale Position einer Pause explizit festzulegen, kann eine Note eingegeben werden, gefolgt vom Befehl `\rest`. Die Pause wird dann an die Stelle gesetzt, wo sich sonst die Note befinden würde.

```
a'4\rest d'4\rest
```



Damit wird die manuelle Formatierung von mehrstimmiger Musik sehr viel einfacher, da die Formatierungsfunktion zur automatischen Auflösung von Zusammenstößen diese Pausen nicht mit einbezieht.

Siehe auch

Programmreferenz: `Rest`.

6.1.10 Überspringen von Zeichen

Eine unsichtbare Pause (auch als „skip“ oder Übersprungung bezeichnet) kann wie eine Note eingegeben werden, die Notationsbezeichnung ist `s`. Man kann aber auch die Dauer extra angeben mit `\skip Dauer`.

```
a4 a4 s4 a4 \skip 1 a4
```



Die `s`-Syntax steht nur im Noten- oder Akkordmodus zur Verfügung. In anderen Situationen, z. B. innerhalb eines Liedtextes, muss `\skip` benutzt werden.

```
<<
  \relative { a'2 a2 }
  \new Lyrics \lyricmode { \skip 2 bla2 }
>>
```



Der Übersprungungsbefehl (`\skip`) ist einfach ein leerer Platzhalter. Durch ihn wird überhaupt nichts gesetzt, auch keine transparenten Objekte.

Die Übersprungung mit `s` hingegen erstellt **Staff** und **Voice** wenn es erforderlich ist, genauso wie Noten und Pausen. Das folgende Beispiel etwa setzt ein leeres Notensystem:

```
{ s4 }
```



Das Fragment `{ \skip 4 }` würde nur eine leere Seite produzieren.

Siehe auch

Programmreferenz: `SkipMusic`.

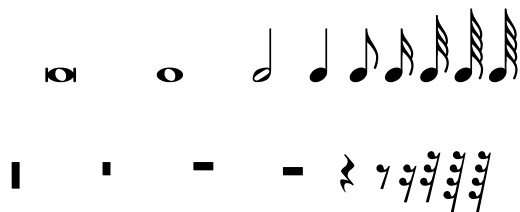
6.2 Rhythmus

Dieser Abschnitt erklärt Rhythmen, Dauern und Takte.

6.2.1 Tondauern

Im Noten-, Akkord- und Liedtextmodus werden Notenlängen (Dauern) durch Zahlen und Punkte notiert: Dauern werden als reziproke Werte geschrieben. Zum Beispiel wird eine Viertelnote mit 4 notiert (weil sie eine 1/4-Note ist), eine halbe Note mit 2 (weil sie eine 1/2-Note ist). Noten, die länger als eine Ganze sind, müssen mit `\longa` (für die Longa, also vier Ganze) und `\breve` (für die Brevis, auch Doppelganze genannt) notiert werden.

```
c'\breve
c'1 c'2 c'4 c'8 c'16 c'32 c'64 c'64
r\longa r\breve
r1 r2 r4 r8 r16 r32 r64 r64
```



Wenn die Dauer hinter einer Notenbezeichnung nicht angegeben ist, wird die Dauer der vorhergehenden Note eingesetzt. Der Standardwert für die erste Note ist eine Viertel.

```
{ a a a2 a4 a a1 a }
```



6.2.2 Punktierung

Um punktierte Notendauern zu erhalten, muss einfach nur ein Punkt (.) hinter die Zahl der Dauer gesetzt werden. Zwei Punkte ergeben eine doppelte Punktierung.

```
a'4 b' c''4. b'8 a'4. b'4.. c''8.
```



Vordefinierte Befehle

Punkte werden normalerweise nach oben verschoben, damit sie die Notenlinien nicht berühren. Das gilt aber nicht für mehrstimmige Passagen. Mit den folgenden Befehlen kann ein anderes Verhalten der Punktierung erreicht werden.

`\dotsUp` (Der Punkt wird nach oben verschoben.), `\dotsDown` (Der Punkt wird nach unten verschoben.), `\dotsNeutral`.

Siehe auch

Programmreferenz: `Dots`, and `DotColumn`.

6.2.3 Andere rhythmische Aufteilungen

Triolen und andere rhythmische Aufteilungen werden aus einem musikalischen Ausdruck erstellt, indem dessen Tondauern mit einem Bruch multipliziert werden.

`\times Bruch musikalischer Ausdruck`

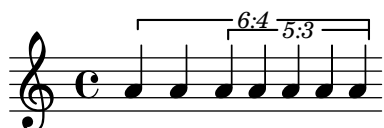
Die Dauer eines *musikalischen Ausdrucks* wird mit dem Bruch multipliziert. Der Nenner des Bruchs wird über den Noten ausgegeben, optional mit einer eckigen Klammer, die die Noten einfasst. Die üblichste Aufteilung ist die Triole, in welcher drei Noten die Länge von zwei haben, der Wert jeder einzelnen Note ist also $\frac{2}{3}$ der notierten Länge.

`g'4 \times 2/3 {c'4 c' c'} d'4 d'4`



Diese Brüche können auch ineinander geschachtelt werden, etwa so:

```
\override TupletNumber #'text = #tuplet-number::calc-fraction-text
\times 4/6 {
  a4 a
  \times 3/5 { a a a a a }
}
```



Vordefinierte Befehle

`\tupletUp`, `\tupletDown`, `\tupletNeutral`.

Übliche Veränderungen der Einstellungen

Der Wert von `tupletSpannerDuration` definiert, wie lange jede Klammer dauert. Mit entsprechender Einstellung kann man beliebig viele Triolen schreiben, aber nur einmal den Befehl `\times` benutzen. Im nächsten Beispiel etwa werden zwei Triolen gedruckt, `\times` aber nur einmal benutzt. Das gilt natürlich auch für alle anderen Brüche.

```
\set tupletSpannerDuration = #(ly:make-moment 1 4)
\times 2/3 { c8 c c c c c }
```



Mehr Information zu `make-moment` findet sich im Abschnitt [Abschnitt 8.4.2 \[Verwaltung der Zeiteinheiten\]](#), Seite 132.

Die Formatierung der Nummer wird durch die Eigenschaft `text` in `TupletNumber` bestimmt. Die Standardeinstellung gibt nur den Nenner aus, aber wenn `TupletNumber` auf den Wert `tuplet-number::calc-fraction-text` gestellt wird, wird *Zähler:Nenner* ausgegeben.

Um gar keine Nummern anzeigen zu lassen, kann folgender Code benutzt werden:

```
\times 2/3 { c8 c c } \times 2/3 { c8 c c }
\override TupletNumber #'transparent = ##t
\times 2/3 { c8 c c } \times 2/3 { c8 c c }
```



Mit der `\tweak`-Funktion können die Zahlen von geschachtelten Brüchen eingestellt werden, die zur gleichen Zeit beginnen. Im Beispiel unten wird mit `\tweak` definiert, dass für die äußere Klammer der Bruch ausgegeben wird, für die innere aber nur der Nenner.

```
\new Staff {
  \tweak #'text #tuplet-number::calc-fraction-text
  \times 4/3 {
    \tweak #'text #tuplet-number::calc-denominator-text
    \times 2/3 { c'8[ c'8 c'8] }
    \times 2/3 { c'8[ c'8 c'8] }
    \times 2/3 { c'8[ c'8 c'8] }
  }
}
```



Im nächsten Beispiel werden `\tweak` und `\override` zusammen verwendet, um die Darstellung der Klammer (`TupletBracket`) zu bestimmen. Mit dem ersten `\tweak` wird die Klammer der äußeren Triole über dem Notensystem platziert. Das zweite `\tweak` platziert die erste der drei inneren Klammern unter dem System (von der Klammer wird hier nur der Nenner des Bruchs dargestellt). Dieses Paar von `\tweak`-Einstellungen wirkt sich jedoch nur auf die äußere und die *erste* innere Klammer aus, weil nur sie zur gleichen Zeit beginnen. Mit dem `\override`-Befehl kann die Richtung der zweiten und dritten `TupletBracket` verändert werden, in diesem Fall wird sie unter das Notensystem geschrieben.

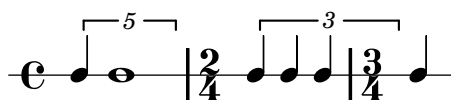
```

\new Staff {
  \tweak #'text #tuplet-number::calc-fraction-text
  \tweak #'direction #up
  \times 4/3 {
    \tweak #'direction #down
    \times 2/3 { c'8[ c'8 c'8] }
    \override TupletBracket #'direction = #down
    \times 2/3 { c'8[ c'8 c'8] }
    \times 2/3 { c'8[ c'8 c'8] }
  }
}

```



Die Klammern können so eingestellt werden, dass sie bis zu einem Taktvorspann oder bis zur nächsten Note reichen.



Siehe auch

Programmreferenz: `TupletBracket`, `TupletNumber` und `TimeScaledMusic`.

6.2.4 Tondauern skalieren

Die Dauer von Längen kann mit einem Bruch multipliziert werden, indem hinter die Note „ $*N/M$ “ (oder „ $*N$ “ wenn $M=1$) geschrieben wird. Das beeinflusst nicht die Erscheinung der Note oder Pause im Druckbild. Die Werte können auch kombiniert werden, etwa „ $*M*N$ “.

Im nächsten Beispiel nehmen die drei ersten Noten genau zwei Schläge ein, aber es wird keine Triolenklammer über ihnen ausgegeben.

```

\time 2/4
a4*2/3 gis4*2/3 a4*2/3
a4 a4 a4*2
b16*4 c4

```



Siehe auch

Abschnitt [Abschnitt 6.2.3 \[Andere rhythmische Aufteilungen\]](#), Seite 71

6.2.5 Taktüberprüfung

Die Taktüberprüfung hilft, Fehler in den Notendauern zu entdecken. Eine Taktüberprüfung wird mit dem Taktstrichsymbol „|“ (Taste AltGr+<) eingegeben. Immer, wenn LilyPond bei der Ausgabe des Notendrucks auf dieses Zeichen stößt, sollte hier in den Noten auch ein Taktstrich erscheinen. Wenn das nicht der Fall ist, wird eine Warnung ausgegeben. Im nächsten Beispiel resultiert die zweite Taktüberprüfung in einer Fehlermeldung.

```
\time 3/4 c2 e4 | g2 |
```

Taktüberprüfungen können auch in Liedtexten verwendet werden:

```
\lyricmode {
  \time 2/4
  Twin -- kle | Twin -- kle
}
```

Eine Taktüberprüfung gilt als nicht bestanden, wenn die Notenwerte nicht stimmen. Besonders in mehrstimmiger komplizierter Musik können solche falschen Notenwerte die ganze Partitur durcheinander bringen. Es lohnt sich also, die Fehlersuche damit zu beginnen, nicht bestandene Taktüberprüfungen zu kontrollieren.

Es ist auch möglich, die Bedeutung des Symbols | umzudefinieren. Das geschieht, indem man der Pipe (`pipeSymbol`) einen musikalischen Ausdruck zuweist:

```
pipeSymbol = \bar "||"
```

```
{ c'2 c' | c'2 c' }
```



6.2.6 Taktnummerüberprüfung

Wenn man größere Musikstücke kopiert, kann es hilfreich sein, wenn LilyPond überprüft, ob die Taktnummer, in der Sie gerade kopieren, mit der des Originalen übereinstimmt. Das kann mit dem Befehl `\barNumberCheck` folgenderweise überprüft werden:

```
\barNumberCheck #123
```

Eine Warnung wird ausgegeben, wenn der interne Zähler `currentBarNumber` von LilyPond nicht mit dem Wert 123 übereinstimmt.

6.2.7 Automatische Aufteilung von Noten

Lange Noten können automatisch in übergebundene Noten aufgeteilt werden. Dieses Verhalten erreicht man, indem der `Note_heads_engraver` mit dem `Completion_heads_engraver` ausgetauscht wird. Im nächsten Beispiel werden Noten, die über die Taktlinie dauern, aufgeteilt und übergebunden.

```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
} {
  c2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2
}
```





Dieser Notationsformatierer (eng. engraver) teilt alle Noten auf, die über eine Taktlinie dauern und fügt Bindebögen hinzu. Er kann unter Anderem dann nützlich sein, wenn man komplexe Partituren auf Fehler überprüfen möchte: Wenn die Takte nicht vollständig gefüllt sind, zeigt die Überbindung genau an, wie viele Notenwerte noch in dem jeweiligen Takt fehlen.

Wenn Sie wollen, dass auch Zeilenumbrüche an den Stellen, an denen automatisch Noten aufgeteilt wurden, stattfinden, müssen Sie auch den Formatierer `Forbid_line_break_engraver` mit dem `\remove`-Befehl entfernen.

Fehler

Nicht alle Notenwerte (besonders wenn sie andere rhythmische Aufteilungen beinhalten) können exakt durch normale Noten und Punktierungen wiedergegeben werden. Der Engraver setzt aber trotzdem keine Triolen etc.

`Completion_heads_engraver` wirkt sich nur auf Noten aus; Pausen werden nicht aufgeteilt.

Siehe auch

Programmreferenz: `Completion_heads_engraver`.

6.3 Mehrstimmigkeit

Polyphonie bedeutet in der musikalischen Terminologie das Vorhandensein von mehr als einer (eigenständigen) Stimme in einem Stück. Für LilyPond bedeutet es aber das Vorhandensein von mehr als einer Stimme pro System.

6.3.1 Akkorde

Ein Akkord wird notiert, indem die zu ihm gehörenden Tonhöhen zwischen spitze Klammern (< und > gesetzt werden. Auf einen Akkord kann eine Dauer-Angabe folgen, genauso wie bei einfachen Noten.

`<c e g>4 <c>8`



Siehe [Abschnitt 7.2 \[Akkordbezeichnungen\]](#), [Seite 117](#) für mehr Information.

6.3.2 Hälse

Immer, wenn das Programm eine Note findet, wird automatisch ein Notenhals (`Stem`) -Objekt erzeugt. Auch für ganze Noten und Pausen werden sie erzeugt, aber unsichtbar gemacht.

Vordefinierte Befehle

`\stemUp` (Hälse nach oben), `\stemDown` (Hälse nach unten), `\stemNeutral` (Hälse je nach Notensposition).

Übliche Veränderungen der Einstellungen

Um die Richtung der Hälse zu ändern, können die Befehle

```
a4 b c b
\override Stem #'neutral-direction = #up
a4 b c b
\override Stem #'neutral-direction = #down
a4 b c b
```

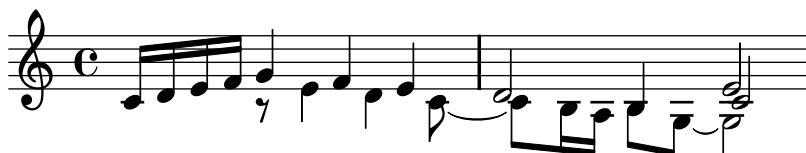


benutzt werden.

6.3.3 Einfache Mehrstimmigkeit

Die einfachste Weise, Abschnitte mit mehr als einer Stimme pro Notensystem zu notieren, ist es, jede Stimme als eine Sequenz zu notieren (innerhalb der Klammern {...}) und dann die beiden Klammern simultan zu kombinieren, indem sie mit \\\ getrennt werden.

```
\new Staff \relative c' {
  c16 d e f
  <<
    { g4 f e | d2 e2 } \\\
    { r8 e4 d c8 ~ | c b16 a b8 g ~ g2 } \\\
    { s2. | s4 b4 c2 }
  >>
}
```



Dieser Trenner veranlasst, dass Stimmen (Voice) -Kontexte² angelegt werden. Sie tragen die Namen "1", "2" usw. In jedem dieser Kontexte wird die Richtung von Bögen, Hälse usw. entsprechend angepasst.

Diese Stimmen sind alle unabhängig von der Stimme, in der die Noten außerhalb der << \\\ >>-Konstruktion notiert sind. Das sollte man berücksichtigen, wenn man auf Stimmen-Ebene Veränderungen vornimmt. Das bedeutet gleichzeitig auch, dass Legato- und Bindebögen nicht in eine << \\\ >>-Umgebung hinein- noch aus hier hinaus ragen können. Die parallelen Notenabschnitte aus unterschiedlichen << \\\ >>-Umgebungen hingegen gehören der gleichen Stimme an. Hier noch einmal das gleiche Beispiel mit unterschiedlichen Notenköpfen für jede Stimme. Die Veränderung der Notenköpfe in der Hauptstimme hat keine Auswirkung auf die Stimmen innerhalb der << \\\ >>-Umgebungen, und die Veränderung der Notenköpfe für die untere Stimme setzt sich fort in der zweiten << \\\ >>-Umgebung. Hier ist auch eine Note über die Taktgrenze hinweg an die untere Stimme der zweiten Polyphonie-Umgebung angebunden.

```
\new Staff \relative c' {
  \override NoteHead #'style = #'cross
```

² Polyphone Stimmen werden in anderen Programmen teilweise als „layers“ (Schichten) bezeichnet.

```

c16 d e f
<<
  { g4 f e } \\
  { \override NoteHead #'style = #'triangle
    r8 e4 d c8 ~ }
>> |
<<
  { d2 e2 } \\
  { c8 b16 a b8 g ~ g2 } \\
  { \override NoteHead #'style = #'slash s4 b4 c2 }
>>
}

```



Polyphonie verändert nicht das Verhältnis der Noten innerhalb einer `\relative { }`-Umgebung. Jede Note wird weiterhin errechnet aus der direkt vorhergehenden.

```
\relative { NoteA << NoteB \\ NoteC >> NoteD }
```

`NoteC` ist relativ zu `NoteB`, nicht `NoteA`; `NoteD` ist relativ zu `NoteC`, nicht `NoteB` oder `NoteA`.

6.3.4 Stimmen explizit beginnen

Voice-Kontexte können auch manuell innerhalb eines `<< >>`-Abschnittes initiiert werden. Mit den Befehlen `\voiceOne` bis hin zu `\voiceFour` kann jeder Stimme entsprechendes Verhalten von vertikaler Verschiebung und Richtung von Hälsen und anderen Objekten hinzugefügt werden.

Genauer gesagt,

```
<< \upper \\ \lower >>
```

entspricht

```

<<
  \new Voice = "1" { \voiceOne \upper }
  \new Voice = "2" { \voiceTwo \lower }
>>

```

Der `\voiceXXX`-Befehl setzt die Richtung von Hälsen, Bögen, Artikulationszeichen, Text, Punktierungen und Fingersätzen. `\voiceOne` und `\voiceThree` lassen diese Objekte nach oben zeigen, `\voiceTwo` und `\voiceFour` dagegen lassen sie abwärts zeigen. Der Befehl `\oneVoice` stellt wieder auf das normale Verhalten um.

Ein Ausdruck, der direkt innerhalb einer `<< >>`-Umgebung auftritt, gehört der Hauptstimme an. Das ist nützlich, wenn zusätzliche Stimme auftreten, während die Hauptstimme sich fortsetzt. Hier also eine bessere Version des Beispiels aus dem vorigen Abschnitt. Die Kreuz-Notenköpfe zeigen, dass die Hauptstimme sich jetzt in einem einzigen Stimmen (`voice`)-Kontext befindet.

```

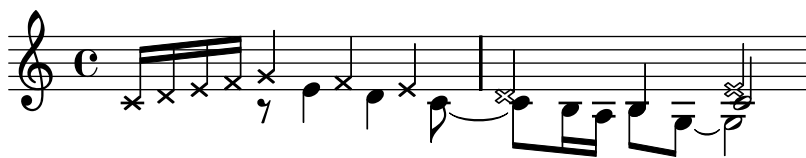
\new Staff \relative c' {
  \override NoteHead #'style = #'cross
  c16 d e f
  \voiceOne
  <<
    { g4 f e | d2 e2 }

```

```

\new Voice="1" { \voiceTwo
  r8 e4 d c8 ~ | c8 b16 a b8 g ~ g2
  \oneVoice
}
\new Voice { \voiceThree
  s2. | s4 b4 c2
  \oneVoice
}
>>
\oneVoice
}

```

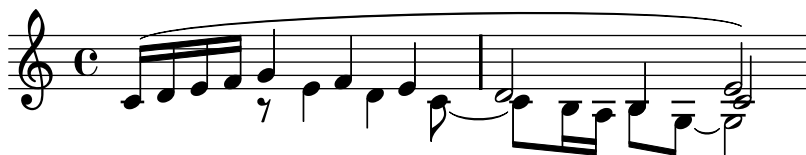


Und mit der richtigen Definition der Stimmen kann die Melodie auch übergebunden werden.

```

\new Staff \relative c' {
  c16^( d e f
  \voiceOne
  <<
    { g4 f e | d2 e2) }
  \context Voice="1" { \voiceTwo
    r8 e4 d c8 ~ | c8 b16 a b8 g ~ g2
    \oneVoice
  }
  \new Voice { \voiceThree
    s2. s4 b4 c2
    \oneVoice
  }
  >>
  \oneVoice
}

```



Indem man den `\-`-Trenner vermeidet, gelingt es auch, mehrstimmige Abschnitte ineinander zu schachteln, was in manchen Fällen die bessere und natürlichere Lösung sein kann.

```

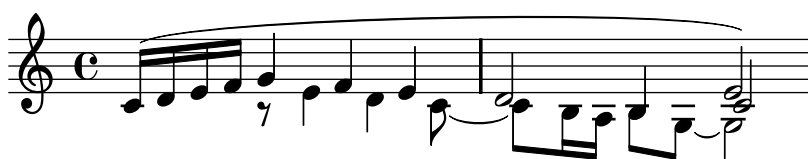
\new Staff \relative c' {
  c16^( d e f
  \voiceOne
  <<
    { g4 f e | d2 e2) }
  \context Voice="1" { \voiceTwo
    r8 e4 d c8 ~ |
    <<

```

```

      {c8 b16 a b8 g ~ g2}
      \new Voice { \voiceThree
        s4 b4 c2
        \oneVoice
      }
    >>
    \oneVoice
  }
  >>
  \oneVoice
}

```



In manchen Fällen von sehr komplexer polyphoner Musik können noch mehr Stimmen benötigt werden, um Zusammenstöße zwischen Noten zu vermeiden. Zusätzliche Stimmen werden durch einen neuen Bezeichner erstellt, wie das nächste Beispiel zeigt.

```

voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)

\relative c''' <<
  { \voiceOne g4 ~ \stemDown g32[ f( es d c b a b64 )g] } \\\
  { \voiceThree b4} \\\
  { \voiceFive d,} \\\
  { \voiceTwo g,}
>>

```



6.3.5 Auflösung von Zusammenstößen

Normalerweise werden Notenköpfe mit einer unterschiedlichen Anzahl von Punktierungen nicht verschmolzen, aber wenn die Objekt-Eigenschaft `merge-differently-dotted` in ein Notenkollisions (`NoteCollision`)-Objekt gesetzt wird, werden sie zusammengefasst.

```

\new Voice << {
  g8 g8
  \override Staff.NoteCollision
    #'merge-differently-dotted = ##t
  g8 g8
} \\\ { g8.[ f16] g8.[ f16] } >>

```



Auf gleiche Art können auch Halbe mit Achteln vereinigt werden, indem `merge-differently-headed` eingesetzt wird:

```
\new Voice << {
  c8 c4.
  \override Staff.NoteCollision
    #'merge-differently-headed = ##t
  c8 c4. } \ { c2 c2 } >>
```



`merge-differently-headed` und `merge-differently-dotted` wirken sich allerdings nur auf Noten mit unterschiedlich gerichteten Hälsen aus (wie etwa Stimme 1 und 2).

LilyPond verschiebt auch Pausen vertikal, die einem Hals gegenüber stehen:

```
\new Voice << c''4 \ { r4 } >>
```



Wenn drei oder mehr Noten in der selben Spalte angeordnet werden, kann `merge-differently-headed` nicht mehr erfolgreich die Noten vereinen, die ineinander gesetzt werden müssten. Damit die Vereinigung funktioniert, muss der Befehl `\shift` vor die Note gesetzt werden, auf die er Auswirkung hat. Im ersten Takt des folgenden Beispiels funktioniert `merge-differently-headed` nicht (der Notenkopf der Halben ist schwarz). Im zweiten Takt wurde `\shiftOn` eingefügt, um das obere g aus der Spalte zu rücken, und das Vereinigen funktioniert wie gewünscht.

```
\override Staff.NoteCollision #'merge-differently-headed = ##t
<<
  { d='2 g2 } \
  { \oneVoice d='8 c8 r4 e,8 c'8 r4 } \
  { \voiceFour e,,2 e'2}
>>
<<
  { d='2 \shiftOn g2 } \
  { \oneVoice d='8 c8 r4 e,8 c'8 r4 } \
  { \voiceFour e,,2 e'2}
>>
```



Vordefinierte Befehle

`\oneVoice`, `\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`.

`\shiftOn`, `\shiftOnn`, `\shiftOnnn`, `\shiftOff`: Diese Befehle definieren den Grad, mit welchem Noten der aktuellen Stimmen verschoben werden sollen. Die äußeren Stimmen (normalerweise 1 und 2) haben den Befehl `\shiftOff`, die inneren dagegen (drei und vier) den Befehl `\shiftOn`. `\shiftOnn` und `\shiftOnnn` stellen weitere Verschiebungsebenen dar.

Wenn LilyPond selber keine Lösung bieten kann, können die Eigenschaft `force-hshift` des `NoteColumn`-Objektes (siehe unten) sowie Pausen mit definierter Tonhöhe eingesetzt werden, um Satzentscheidungen des Programmes zu überschreiben.

```
\relative <<
{
  <d g>
  <d g>
} \ {
  <b f'>
  \once \override NoteColumn #'force-hshift = #1.7
  <b f'>
} >>
```



Siehe auch

Programmreferenz: Die Objekte, die für Auflösung von Zusammenstößen zuständig sind, sind `NoteCollision` und `RestCollision`.

Fehler

Wenn `merge-differently-headed` mit einer nach oben gerichteten Achtel oder kleineren Note verwendet wird, und die nach unten gerichtete Note ist eine Halbe, bekommt die Achtel die falsche Richtung gesetzt.

Es gibt keine Unterstützung für Cluster, in denen die gleiche Note mit unterschiedlichen Vorzeichen im selben Akkord auftritt. In diesem Fall sollte man eine enharmonische Transkription benutzen oder die spezielle Cluster-Notationsweise, siehe [Abschnitt 8.4.4 \[Cluster\]](#), [Seite 132](#).

6.4 Notation innerhalb von Systemen

Dieser Abschnitt zeigt die Notation von Symbolen, die auf Systemebene wirken, wie Tonartvorzeichen, Schlüssel oder Taktangaben.

6.4.1 Notenschlüssel

Der Schlüssel zeigt eine bestimmte Systemlinie an und markiert die Tonhöhe, mit der sie korrespondiert. Ein Schlüssel wird mit dem `\clef`-Befehl gesetzt.

```
{ c'2 \clef alto g'2 }
```



Unterstützt sind folgende Schlüssel:

Schlüssel	Lage
treble (Violinschlüssel)	G-Schlüssel auf der zweiten Linie
alto, C (Bratschenschlüssel)	C-Schlüssel auf der dritten Linie
tenor (Tenorschlüssel)	C-Schlüssel auf der vierten Linie
bass, F (Bassschlüssel)	F-Schlüssel auf der vierten Linie
french (Franz. Violinschlüssel)	G-Schlüssel auf der ersten Linie
soprano (Sopranschlüssel)	C-Schlüssel auf der ersten Linie
mezzosoprano (Mezzosopranschlüssel)	G-Schlüssel auf der zweiten Linie
baritone (Baritonschlüssel)	C-Schlüssel auf der fünften Linie
varbaritone (Bariton-F-Schlüssel)	F-Schlüssel auf der dritten Linie
subbass (Kontrabassschlüssel)	F-Schlüssel auf der fünften Linie
percussion	Schlagzeugschlüssel
tab	Tabulaturschlüssel

Indem `_8` oder `~8` an die jeweilige Schlüsselbezeichnung angehängt wird, wird der Schlüssel um eine Oktave nach oben oder unten transponiert, mit `_15` oder `~15` um zwei Oktaven. Die Schlüsselbezeichnung muss in Anführungszeichen gesetzt werden, wenn sie Unterstriche oder Zahlen enthält, siehe Beispiel:

```
\clef "G_8" c4
```



Übliche Veränderungen der Einstellungen

Dem Befehl `\clef "treble_8"` entspricht die entsprechende separate Einstellung von `clefGlyph`, `clefPosition` (womit y- und x-Position des Schlüssel bestimmt werden), `middleCPosition` und `clefOctavation`. Ein Schlüssel wird immer dann ausgegeben, wenn eine dieser Eigenschaften sich ändert. Im nächsten Beispiel werden Möglichkeiten gezeigt, die Eigenschaften manuell zu setzen.

```
{
  \set Staff.clefGlyph = #"clefs.F"
  \set Staff.clefPosition = #2
  c'4
  \set Staff.clefGlyph = #"clefs.G"
  c'4
  \set Staff.clefGlyph = #"clefs.C"
  c'4
  \set Staff.clefOctavation = #7
}
```

```

c'4
\set Staff.clefOctavation = #0
\set Staff.clefPosition = #0
c'4
\clef "bass"
c'4
\set Staff.middleCPosition = #4
c'4
}

```



Siehe auch

Handbuch: [Abschnitt 6.5.7 \[Verzierungen\]](#), Seite 98.

Programmreferenz: `Clef`.

6.4.2 Tonartbezeichnung

Die Vorzeichen zeigen die Tonart an, in welcher ein Stück notiert ist. Es handelt sich um eine Anzahl von Alterationszeichen (Kreuzen oder Bs) am Beginn jedes Notensystems.

Das Setzen und Ändern von Tonarteinstellungen wird mit dem `\key`-Befehl vorgenommen.

```
\key Tonhöhe Art
```

Der Wert *Art* sollte entweder `\major` oder `\minor` sein, um Moll oder Dur der *Tonhöhe* zu erhalten. Es können auch Modusbezeichnungen für Kirchentonarten verwendet werden: `\ionian` (Ionisch), `\locrian` (Locrisch), `\aeolian` (Aeolisch), `\mixolydian` (Mixolydisch), `\lydian` (Lydisch), `\phrygian` (Phrygisch) und `\dorian` (Dorisch).

Dieser Befehl ändert die Kontexteigenschaft `Staff.keySignature`. Vorzeichen, die nicht dem Standard entsprechen, können manuell mit dieser Eigenschaft eingegeben werden.

Versetzungszeichen und Vorzeichen können am Anfang etwas verwirrend sein, da unveränderte Noten je nach Tonart ein Auflösungszeichen bekommen können (Beispiel). Mehr Information in den Abschnitten [Abschnitt 6.1.2 \[Versetzungszeichen\]](#), Seite 63 oder [Abschnitt 2.2.2 \[Versetzungszeichen und Tonartbezeichnung \(Vorzeichen\)\]](#), Seite 18.

```

\key g \major
f1
fis

```



Übliche Veränderungen der Einstellungen

Ein Auflösungszeichen wird gesetzt, um vorhergehende Vorzeichen rückgängig zu machen. Das kann aber unterdrückt werden, indem die `Staff.printKeyCancellation`-Eigenschaft angepasst wird.

```
\key d \major
a b cis d
\key g \minor
a bes c d
\set Staff.printKeyCancellation = ##f
\key d \major
a b cis d
\key g \minor
a bes c d
```



Siehe auch

Programmreferenz: `KeyCancellation`, `KeySignature`.

6.4.3 Taktangabe

Taktangaben zeigen das Metrum eines Stückes an: eine regelmäßige Folge von betonten und unbetonten Zeiten. Es wird angezeigt als ein Bruch vor Beginn der Noten.

Die Taktangabe wird mit dem `\time`-Befehl gesetzt.

```
\time 2/4 c'2 \time 3/4 c'2.
```



Übliche Veränderungen der Einstellungen

Das Symbol, das angezeigt wird, kann durch die `style`-Eigenschaft angepasst werden. Wenn man es auf den Wert `#'()` setzt, wird auch für 4/4- und 2/2-Takte ein Bruch angezeigt.

```
\time 4/4 c'1
\time 2/2 c'1
\override Staff.TimeSignature #'style = #'()
\time 4/4 c'1
\time 2/2 c'1
```

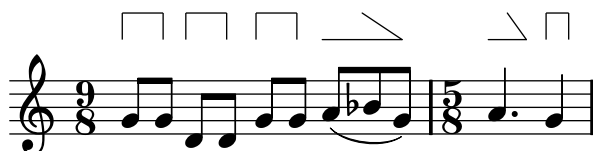


Es gibt noch sehr viel mehr Optionen für das Layout. Siehe den Abschnitt [Abschnitt 7.7.6 \[Taktangaben der alten Musik\]](#), Seite 120 für weitere Beispiele.

Der Befehl `\time` stellt die Eigenschaften `timeSignatureFraction`, `beatLength` und `measureLength` im `Timing`-Kontext ein, der normalerweise zu den Eigenschaften aller System (Score-Ebene) gehört. Die Eigenschaft `measureLength` bestimmt, wo Taktlinien eingefügt werden, und wie automatische Balken gesetzt werden. Ein Verändern der Eigenschaften von `timeSignatureFraction` gibt das neue Symbol an dieser Stelle aus.

Auf weitere Optionen kann man über die Scheme-Funktion `set-time-signature` zugreifen. In Kombination mit `Measure_grouping_engraver` werden hiermit `MeasureGrouping`-Zeichen erzeugt. Solche Zeichen erleichtern es, rhythmisch komplexe moderne Musik zu lesen. Im nächsten Beispiel ist der 9/8-Takt in 2, 2, 2 und 3 Achtel unterteilt. Das wird im dritten Argument an `set-time-signature` weitergegeben (2 2 2 3).

```
\score {
  \relative c'' {
    #(set-time-signature 9 8 '(2 2 2 3))
    g8[ g] d[ d] g[ g] a8[( bes g)] |
    #(set-time-signature 5 8 '(3 2))
    a4. g4
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```



Siehe auch

Programmreferenz: `TimeSignature` und `Timing_translator`.

Beispiele: `'input/test/compound-time.ly'`.

Fehler

Automatische Balken richten sich nicht nach den Taktunterteilungen, die mit `set-time-signature` erzeugt werden.

6.4.4 Auftakte

Verkleinerte Takte, wie etwa ein Auftakt, werden wie folgt notiert:

```
\partial 16*5 c16 cis d dis e | a2. c,4 | b2
```



Die Syntax für den Befehl lautet:

```
\partial Dauer
```

wobei *Dauer* eine rhythmische Länge ist, die vor dem nächsten Taktstrich eingefügt wird.

Das wird intern übersetzt nach:

```
\set Timing.measurePosition = -Länge der Dauer
```

Die Eigenschaft `measurePosition` enthält eine rationale Zahl, die darstellt, wie groß der Abstand zum Taktanfang ist. Deshalb ist sie eine negative Zahl; `\partial 4` wird also intern übersetzt zu: „Eine Viertel bleibt übrig vom ganzen Takt.“

Fehler

Dieser Befehl berücksichtigt keine Verzierungen/Vorschläge am Anfang der Noten. Wenn ein Stück mit einem Vorschlag anfängt, muss der Befehl `\partial nach` dem Vorschlag kommen:

```
\grace f16
\partial 4
g4
a2 g2
```



`\partial` ist nur für den Anfang eines Stückes vorgesehen. Wenn der Befehl innerhalb eines Stückes verwendet wird, können seltsame Warnungen auftreten.

6.4.5 Taktlinien

Taktlinien trennen die Takte voneinander, werden aber auch verwendet, um Wiederholungen anzuzeigen. Normalerweise werden sie automatisch eingefügt. Zeilenumbrüche können nur an Taktlinien stattfinden.

Besondere Taktlinien-Arten können mit dem `\bar`-Befehl erzwungen werden.

```
c4 \bar " |: " c4
```



Folgende Taktlinienarten sind vorhanden:





Als letztes ist "||:" notiert, das sich ähnlich wie ":" verhält. Es gibt jedoch nur an Zeilenenden eine doppelte Taktlinie aus und fängt die Wiederholungslinie erst in der nächsten Zeile an.

Um einen Zeilenumbruch an einer Stelle zu erlauben, wo keine sichtbare Taktlinie ist, kann man

```
\bar ""
```

benutzen. Damit wird eine unsichtbare Taktlinie an dieser Stelle eingefügt und damit ein Zeilenumbruch erlaubt (ohne dass sich die Anzahl der Takte erhöhen würde).

In Partituren mit vielen Systemen wird ein `\bar`-Befehl in einem System automatisch auf alle anderen Systeme angewendet. Die resultierenden Taktlinien sind miteinander verbunden innerhalb einer Gruppe (`StaffGroup`) oder einem Klaviersystem (`PianoStaff` bzw. (`GrandStaff`).

```
<<
  \new StaffGroup <<
    \new Staff {
      e'4 d'
      \bar "||"
      f' e'
    }
    \new Staff { \clef bass c4 g e g }
  >>
  \new Staff { \clef bass c2 c2 }
>>
```



Übliche Veränderungen der Einstellungen

Der Befehl `\bar Taktart` ist eine Kurzform von: `\set Timing.whichBar = Taktart`. Immer, wenn `whichBar` auf einen Wert gesetzt wird, wird eine Taktlinie dieses Typs erzeugt.

Eine Taktlinie wird auch durch Setzen der `whichBar`-Eigenschaft erzeugt. Am Anfang eines Taktes wird sie auf den Wert von `Timing.defaultBarType` gesetzt. Der Inhalt des `repeatCommands`-Befehls wird benutzt, um Standardtaktlinien zu überschreiben.

Sie sollten jedoch Wiederholungen mit dem `\repeat`-Befehl erzeugen. Siehe Abschnitt [Abschnitt 6.7 \[Wiederholungszeichen\]](#), Seite 110.

Siehe auch

Im Handbuch: [Abschnitt 6.7 \[Wiederholungszeichen\]](#), Seite 110, [Abschnitt 6.4.7 \[Klammern am Systemanfang\]](#), Seite 88.

Programmreferenz: `BarLine` (auf Partitur (`Staff`)-Ebene erstellt), `SpanBar` (zwischen Systemen).

6.4.6 Musik ohne Metrum

Taktlinien und Taktzahlen werden automatisch erzeugt. Für Musik ohne Meter hingegen (etwa Kadenzen) ist das allerdings nicht erwünscht. Mit den Befehlen `\cadenzaOn` und `\cadenzaOff` kann dieses Verhalten ausgeschaltet und wieder angeschaltet werden.

```
c4 d e d
\cadenzaOn
c4 c d8 d d f4 g4.
\cadenzaOff
\bar "|"
d4 e d c
```



Fehler

LilyPond fügt Zeilen- und Seitenumbrüche nur an einer Taktlinie ein. Wenn die Kadenz nicht vor einem Umbruch endet, müssen Sie selber unsichtbare Taktlinien

```
\bar ""
```

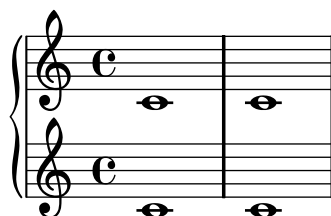
einfügen, um anzuzeigen, wo umgebrochen werden darf.

6.4.7 Klammern am Systemanfang

Viele Partituren bestehen aus mehr als einem Notensystem. Diese Systeme können auf vier unterschiedliche Arten verbunden werden:

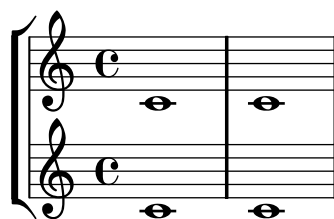
- Die Gruppe wird mit einer Klammer an der linken Seite geöffnet und die Taktlinien sind verbunden. Das ist der Klaviersystem (`GrandStaff`)-Kontext.

```
\new GrandStaff
\relative <<
  \new Staff { c1 c }
  \new Staff { c c }
>>
```



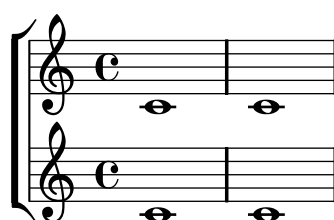
- Die Gruppe beginnt mit einer Klammer und die Taktlinien sind verbunden. Dieses Verhalten erzeugt der Stimmgruppen (`StaffGroup`)-Kontext.

```
\new StaffGroup
\relative <<
  \new Staff { c1 c }
  \new Staff { c c }
>>
```



- Die Gruppe beginnt mit einer Klammer, aber die Taktlinien sind nicht miteinander verbunden. Das wird mit dem Chorsystem (`ChoirStaff`)-Kontext erreicht.

```
\new ChoirStaff
\relative <<
  \new Staff { c1 c }
  \new Staff { c c }
>>
```



- Die Gruppe beginnt mit einer vertikalen Linie. Taktlinien sind nicht verbunden. Das ist die Standardeinstellung für eine Partitur.

```
\relative <<
  \new Staff { c1 c }
  \new Staff { c c }
>>
```



Siehe auch

Die Definition der Taktlinien am Beginn jedes Systems werden mit den Befehlen `SystemStartBar`, `SystemStartBrace` und `SystemStartBracket` festgelegt. Nur einer dieser drei Typen wird in jedem Kontext erstellt, und dieser Typ wird durch die `systemStartDelimiter`-Eigenschaft bestimmt.

Übliche Veränderungen der Einstellungen

Anfangsklammern können tief einander verschachtelt werden.

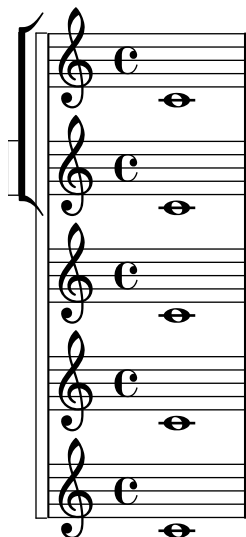
```
\new StaffGroup
\relative <<
  \set StaffGroup.systemStartDelimiterHierarchy
    = #'(SystemStartSquare (SystemStartBracket a (SystemStartSquare b)) d)
  \new Staff { c1 }
```

```

\new Staff { c1 }
\new Staff { c1 }
\new Staff { c1 }
\new Staff { c1 }

```

```
>>
```



6.4.8 Das Notensystem

Noten, Dynamikzeichen usw. werden auf den Notenlinien angeordnet, die sich zu einem Notensystem zusammenfassen lassen. Das Programm LilyPond zeichnet diese Linien durch ein spezielles graphisches Objekt, `staff symbol` (engl. „staff“ = Notensystem) genannt.

Dieses Objekt kann bezüglich seiner Eigenschaften, wie Anzahl, Dicke und Abstand der Linien verändert werden. Das wird gezeigt in den Beispieldateien ‘`input/test/staff-lines.ly`’ und ‘`input/test/staff-size.ly`’.

Zusätzlich können Systeme beliebig begonnen und beendet werden. Das geschieht mit den Befehlen `\startStaff` und `\stopStaff`.

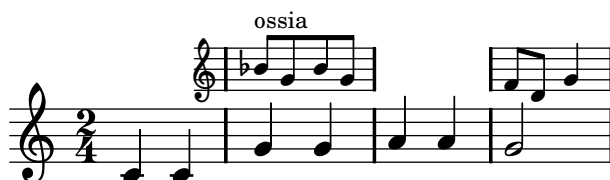
```

b4 b
\override Staff.StaffSymbol #'line-count = 2
\stopStaff \startStaff
b b
\revert Staff.StaffSymbol #'line-count
\stopStaff \startStaff
b b

```



Kombiniert mit verkleinerten Systemen, kann man diese Funktion etwa benutzen, um Ossia-Abschnitte zu notieren. Siehe das Beispiel:



Siehe auch

Programmreferenz: `StaffSymbol`.

Beispiele: `'input/test/staff-lines.ly'`, `'input/test/ossia.ly'`, `'input/test/staff-size.ly'`, `'staff/staff-line-positions.ly'`

6.4.9 Musik parallel notieren

Noten für mehrere Stimmen können verschachtelt notiert werden:

```
\parallelMusic #'(voiceA voiceB) {
  r8      g'16[ c''] e''[ g' c'' e''] r8      g'16[ c''] e''[ g' c'' e''] |
  c'2                                           c'2 |
  r8      a'16[ d''] f''[ a' d'' f''] r8      a'16[ d''] f''[ a' d'' f''] |
  c'2                                           c'2 |
}
\new StaffGroup <<
  \new Staff \new Voice \voiceA
  \new Staff \new Voice \voiceB
>>
```



Das funktioniert ziemlich gut für Klaviernoten:

```
music = {
  \key c \major
  \time 4/4
  \parallelMusic #'(voiceA voiceB voiceC voiceD) {
    % Bar 1
    r8 g'16[ c''] e''[ g' c'' e''] r8 g'16[ c''] e''[ g' c''
e''] |
    c'2                                           c'2 |
    r8 a16[ d'] f'[ a d' f']          r8 a16[ d'] f'[ a d' f'] |
    c2                                           c2 |

    % Bar 2
    a'8 b'          c'' d''          e'' f''          g'' a'' |
    d'4             d'              d'              d' |
```

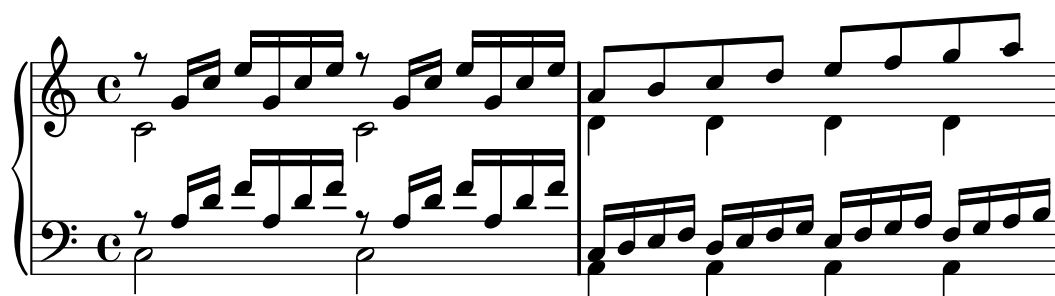
```

c16 d e f    d e f g    e f g a    f g a b |
a,4          a,4          a,4          a,4 |

% Bar 3 ...
}
}

\score {
  \new PianoStaff <<
    \music
    \new Staff <<
      \voiceA \
      \voiceB
    >>
    \new Staff {
      \clef bass
      <<
        \voiceC \
        \voiceD
      >>
    }
  >>
}

```



6.5 Noten verbinden

Dieser Abschnitt stellt Eigenschaften dar, die Gruppen von Noten betreffen.

6.5.1 Bindebögen

Ein Bindebogen verbindet zwei benachbarte Noten der selben Tonhöhe. Als Resultat wird die Dauer der Notenlänge verlängert. Bindebögen dürfen nicht mit Legatobögen verwechselt werden, durch die die Vortragsart bezeichnet wird, noch mit Phrasierungsbögen, die musikalische Phrasen anzeigen. Eine Bindebogen wird mit der Tilde ~ (AltGr++) notiert.

```
e' ~ e' <c' e' g'> ~ <c' e' g'>
```



Wenn ein Bindebogen an einen Akkord gehängt wird, werden alle Noten dieses Akkordes übergebunden. Wenn kein Notenkopf passt, wird auch kein Bogen erzeugt. Noten in Akkorden können auch einzeln übergebunden werden, indem sie innerhalb des Akkordes hinter die entsprechende Note geschrieben werden.

<c~ e g~ b> <c e g b>



Ein Bindebogen ist nur eine andere Art, die Notendauer zu verlängern, ähnlich wie die Punktierung. Im nächsten Beispiel sind zwei Arten gezeigt, die gleiche Notenlänge zu notieren:



Bindebögen werden verwendet, wenn die Note entweder über die Taktgrenze hinausragt, oder wenn Punktierung nicht benutzt werden kann, um die Verlängerung anzuzeigen. Wenn man Überbindungen verwendet, sollten größere Notenwerte an die Unterteilungen des Taktes angepasst werden.



Wenn sehr viele Noten über die Taktgrenzen hinüber angebonden werden müssen, ist es oft einfacher, die automatische Aufteilung von Noten einzusetzen (siehe [Abschnitt 6.2.7 \[Automatische Aufteilung von Noten\]](#), Seite 74). Mit dieser Funktion werden automatisch lange Noten aufgeteilt und über die Taktgrenze übergebunden.

Wenn die zweite Variante einer Wiederholung mit einer Überbindung anfängt, muss der Bindebogen wiederholt werden. Dass geschieht durch den Befehl `\repeatTie`.



Übliche Veränderungen der Einstellungen

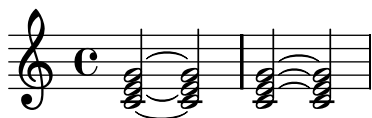
Bindebögen werden teilweise verwendet, um Arpeggien auszuschreiben. In diesem Fall müssen mehrere übergebundene Noten nacheinander erscheinen. Das ist möglich, indem die `tieWaitForNote`-Eigenschaft auf wahr (`##t`) gesetzt wird. Diese Funktion ist auch nützlich, um ein Tremolo an einen Akkord zu binden. Siehe das Beispiel:

```
\set tieWaitForNote = ##t
\grace { c16[~ e~ g]~ } <c, e g>2
\repeat "tremolo" 8 { c32~ c'~ } <c c,>1
e8~ c~ a~ f~ <e' c a f>2
```



Bindebögen können manuell gesetzt werden, indem die `tie-configuration`-Eigenschaft verändert wird. Die erste Zahl zeigt den Abstand von der Mitte des Notensystems in Notenlinienzwischenräumen, die zweite die Richtung (1=nach oben, -1=nach unten).

```
<c e g>2~ <c e g> |
\override TieColumn #'tie-configuration =
  #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
<c e g>~ <c e g> |
```



Vordefinierte Befehle

`\tieUp`, `\tieDown`, `\tieNeutral`, `\tieDotted`, `\tieDashed`, `\tieSolid`.

Siehe auch

Im Handbuch: [Abschnitt 6.2.7 \[Automatische Aufteilung von Noten\]](#), Seite 74.

Programmreferenz: `Tie`.

Fehler

Der Wechsel zwischen Systemen bei aktiver Überbindung produziert keinen gekrümmten Bogen.

Änderung von Schlüssel oder Oktavierung zwischen übergebundenen Noten ist nicht richtig definiert. In diesen Fällen kann es besser sein, einen Legatobogen zu verwenden.

6.5.2 Legatobögen

Ein Legatobogen (engl. slur) zeigt an, dass die Noten *legato* gespielt werden sollen. Er wird mit Klammern hinter den Notenwerten notiert.

```
f( g a) a8 b( a4 g2 f4)
<c e>2( <b d>2)
```



Die Richtung eines Legatobogens kann mit den Befehlen `\slurDIR`, wobei *DIR* entweder *Up*, *Down*, oder *Neutral*, angezeigt werden.

Es gibt aber auch eine Kurzform. Indem `_` oder `^` for die öffnende Klammer gestellt wird, wird die Richtung angegeben.

```
c4_( c) c^( c)
```



Nur ein Legatobogen kann gleichzeitig geschrieben werden. Wenn Sie einen langen Bogen über mehreren kurzen notieren wollen, müssen Sie [Abschnitt 6.5.3 \[Phrasierungsbögen\], Seite 95](#) benutzen.

Übliche Veränderungen der Einstellungen

Manche Komponisten schreiben zwei Legatobögen, um Legatoakkorde zu markieren. Das kann in LilyPond erreicht werden, indem die Eigenschaft `doubleSlurs` gesetzt wird.

```
\set doubleSlurs = ##t
<c e>4 ( <d f> <c e> <d f> )
```



Vordefinierte Befehle

`\slurUp`, `\slurDown`, `\slurNeutral`, `\slurDashed`, `\slurDotted`, `\slurSolid`.

Siehe auch

Programmreferenz: `internals document`, `Slur`.

6.5.3 Phrasierungsbögen

Ein Phrasierungsbogen verbindet Noten und wird verwendet, um einen musikalischen Ausdruck anzuzeigen. Er wird mit den Befehlen `\(` und `\)` eingegeben.

```
\time 6/4 c' \( d( e) f( e) d\)
```



Im typographischen Sinne verhalten sich Phrasierungsbögen genauso wie Legatobögen. Sie werden aber als eigene Objekte behandelt. Ein `\slurUp` hat also keine Auswirkung auf die Phrasierungsbögen, anstelle dessen muss `\phrasingSlurUp`, `\phrasingSlurDown` oder `\phrasingSlurNeutral` benutzt werden.

Es können keine simultanen Phrasierungsbögen gesetzt werden.

Vordefinierte Befehle

`\phrasingSlurUp`, `\phrasingSlurDown`, `\phrasingSlurNeutral`.

Siehe auch

Programmreferenz: `PhrasingSlur`.

6.5.4 Laissez vibrer-Bögen

So genannte „laissez vibrer“-Bögen werden verwendet um anzuzeigen, dass man die Musik ausklingen lassen soll. Sie werden in der Klavier-, Harfen-, anderer Saiteninstrument- und Schlagzeugnotation verwendet. Sie können mit dem Befehl `\laissezVibrer` eingegeben werden.

```
<c f g>\laissezVibrer
```



Siehe auch

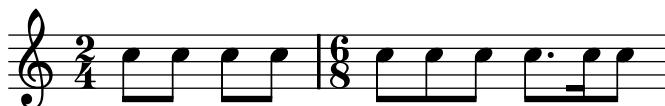
Programmreferenz: `LaissezVibrerTie` `LaissezVibrerTieColumn`

Beispiele: ‘connecting/laissez-vibrer-ties.ly’

6.5.5 Automatische Balken

LilyPond setzt Balken (engl. beam) automatisch.

```
\time 2/4 c8 c c c \time 6/8 c c c c8. c16 c8
```



Wenn diese automatischen Entscheidungen nicht gut genug sind, können die Balken auch explizit eingegeben werden. Es können auch bestimmte Balkenmuster, die sich vom Standard unterscheiden, definiert werden. Hierzu siehe den Abschnitt [Abschnitt 9.1.2 \[Einstellung von automatischen Balken\]](#), [Seite 134](#) für Einzelheiten.

Einzelne Noten können mit dem Befehl `\noBeam` markiert werden, damit sie nicht mit einem Balken versehen werden.

```
\time 2/4 c8 c\noBeam c c
```



Siehe auch

Programmreferenz: `Beam`.

6.5.6 Manuelle Balken

In einigen Fällen kann es nötig sein, den automatischen Algorithmus für die Balken zu überschreiben. Die automatischen Balken werden beispielsweise nicht über Pausen oder Taktlinien hinweg gesetzt. Manuell definierte Balken werden mit den Zeichen `[` und `]` (AltGr+8 bzw. 9) markiert.

```
{
  r4 r8[ g' a r8] r8 g[ | a] r8
}
```



Übliche Veränderungen der Einstellungen

Normalerweise werden die Balken innerhalb eines größeren Balkens automatisch bestimmt. Wenn nötig, kann aber mit den Eigenschaften `stemLeftBeamCount` und `stemRightBeamCount` ein anderes Verhalten erreicht werden. Wenn eine der Eigenschaften gesetzt ist, wird es nur einmal verwendet und dann wieder zum Standard zurück geschaltet.

```
{
  f8[ r16
    f g a]
  f8[ r16
    \set stemLeftBeamCount = #1
    f g a]
}
```



Die Eigenschaft `subdivideBeams` kann benutzt werden, um alle 16tel-Balken oder kleinere Werte zu bestimmten Taktzeiten zu unterteilen. Diese Zeiten werden mit der `beatLength`-Eigenschaft gesetzt.

```
c16[ c c c c c c c]
\set subdivideBeams = ##t
c16[ c c c c c c c]
\set Score.beatLength = #(ly:make-moment 1 8)
c16[ c c c c c c c]
```



Zu mehr Information über `make-moment` siehe [Abschnitt 8.4.2 \[Verwaltung der Zeiteinheiten\]](#), [Seite 132](#).

Zeilenumbrüche sind normalerweise verboten, wenn Balken sich über die Taktlinien erstrecken. Das kann aber durch Setzen von `breakable` verändert werden.

Fehler

Balken mit Hälsen nach oben und unten werden automatisch eingesetzt, wenn ein großer Abstand zwischen Notenköpfen gefunden wird. Die Größe des Wertes kann mit dem `auto-knee-gap`-Objekt eingestellt werden.

Automatisch erstellte Balken zwischen Systemen können nicht zusammen mit automatisch versteckten Systemen verwendet werden. Siehe auch [Abschnitt 8.3.2 \[Systeme verstecken\]](#), Seite 131.

Balken vermeiden nicht andere Objekte, wie etwa Text und Versetzungszeichen.

6.5.7 Verzierungen

Verzierungen sind ausgeschriebene Ornamente. Die üblichste ist der Vorschlag. Er wird durch eine verkleinerte Note mit Schrägstrich und Bogen notiert. Der Vorhalt dagegen ist eine Verzierung, die einen bestimmten Notenwert der Hauptnote für sich beansprucht. Er wird als verkleinerte Note ohne Schrägstrich notiert. Der Vorschlag wird mit dem Befehl `\acciaccatura` und der Vorhalt mit `\appoggiatura` eingegeben.

```
b4 \acciaccatura d8 c4 \appoggiatura e8 d4
\acciaccatura { g16[ f] } e4
```



Bei beiden handelt es sich um spezielle Formen des `\grace` (engl. Verzierung)-Befehl. Wenn dieser Befehl einem musikalischen Ausdruck vorgestellt wird, wird ein neuer Ausdruck geschaffen, der in kleineren Noten gesetzt wird und von der logischen Zeit innerhalb des Taktes keinen Raum beansprucht.

```
c4 \grace c16 c4
\grace { c16[ d16] } c2 c4
```



Anders als `\acciaccatura` oder `\appoggiatura` setzt der `\grace`-Befehl keinen Bogen.

Programmintern wird die Zeitberechnung für Verzierungen in einer zweiten Zähllebene vorgenommen. Jeder Zeitpunkt beinhaltet zwei rationale Zahlen: die eine steht für die logische Zeit, die andere für die „Verzierungszeit“. Das obere Beispiel ist hier mit den entsprechenden Zeitwerten angezeigt:



Die Position von Verzierungen wird zwischen den Systemen synchronisiert. Im folgenden Beispiel sind jeweils zwei Sechzehntel gegen jede Achtel gesetzt:

```
<< \new Staff { e4 \grace { c16[ d e f] } e4 }
      \new Staff { c4 \grace { g8[ b] } c4 } >>
```



Eine Verzierung kann auch auf eine Note folgend gesetzt werden. Dazu wird der `\afterGrace`-Befehl benutzt. Er nimmt zwei Argumente: die Hauptnote und die Verzierungen, die nach der Hauptnote erscheinen sollen.

```
c1 \afterGrace d1 { c16[ d] } c4
```



Damit wird die Verzierung im Abstand von $\frac{3}{4}$ der Länge der Hauptnote gesetzt. Dieser Bruch kann durch Setzen von `afterGraceFraction` verändert werden:

```
#(define afterGraceFraction (cons 7 8))
```

Hier wurde die Position auf das vorletzte Achtel der Notenlänge gesetzt.

Der gleiche Effekt kann auch manuell erreicht werden:

```
\new Voice {
  << { d1^\trill_( }
      { s2 \grace { c16[ d] } } >>
  c4)
}
```



Indem die Dauer der unsichtbaren Note (hier eine Halbe) wird der Abstand zwischen Hauptnote und Verzierung angepasst.

Ein `\grace`-Abschnitt wird nach besonderen Satzregeln gesetzt, um z. B. kleinere Noten zu benutzen und die Richtung der Hälse einzustellen. Veränderungen am Layout müssen also innerhalb der Verzierung gesetzt werden, damit sie auch eine Auswirkung haben.

```
\new Voice {
  \acciaccatura {
    \stemDown
    f16->
    \stemNeutral
  }
  g4
}
```



Diese Einstellungen müssen auch innerhalb der Verzierungsebene wieder rückgängig gemacht werden.

Das Layout der Verzierungsabschnitte kann mit der Funktion `add-grace-property` verändert werden. Im folgenden Beispiel wird die Richtung der Notenhäse neutralisiert, so dass sie nicht unbedingt in die gleiche Richtung zeigen.

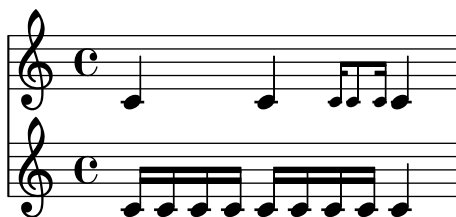
```
\new Staff {
  #(add-grace-property 'Voice 'Stem 'direction '())
  ...
}
```

Eine andere Option ist es, die Variablen `startGraceMusic`, `stopGraceMusic`, `startAcciaccaturaMusic`, `stopAcciaccaturaMusic`, `startAppoggiaturaMusic` und `stopAppoggiaturaMusic` zu vermeiden. Mehr Information findet sich in der Datei `'ly/grace-init.ly'`.

Der Schrägstrich durch den Notenhals der Vorschläge kann auch in anderen Situation erreicht werden mit `\override Stem #'stroke-style = #"grace"`.

Übliche Veränderungen der Einstellungen

Verzierungen könne gezwungen werden, den Hauptnoten entsprechend aufgeteilt zu werden.



Siehe auch

Programmreferenz: `GraceMusic`.

Fehler

Eine Partitur, die mit einem `\grace`-Abschnitt beginnt, benötigt eine explizit gesetzte neue Stimme (`\new Voice`), sonst werden Hauptnote und Verzierung auf verschiedenen Systemen gesetzt.

Die Synchronisation von Verzierungen kann auch zu Überraschungen führen. Auch andere Symbole der Systeme, wie Vorzeichen, Taktlinien usw., werden synchronisiert. Vorsicht ist geboten, wenn nur in bestimmten Systemen Verzierungen vorkommen:

```
<< \new Staff { e4 \bar "|:" \grace c16 d4 }
    \new Staff { c4 \bar "|:" d4 } >>
```



Dem kann abgeholfen werden, indem unsichtbare Verzierungsnoten der selben Länge in die anderen Systeme gesetzt werden. Im obigen Beispiel müsste also

```
\new Staff { c4 \bar "|:" \grace s16 d4 }
```

gesetzt werden.

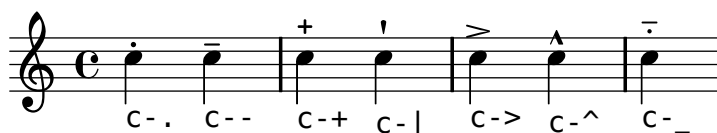
Verzierungsabschnitte sollten nur innerhalb von sequentiellen musikalischen Ausdrücken benutzt werden. Wenn sie ineinandergeschachtelt werden, kann es zu Fehlermeldungen oder Abstürzen kommen.

6.6 Ausdrucksbezeichnungen

Vortragszeichen helfen dem Musiker, mehr Ausdruck in die Musik zu legen.

6.6.1 Artikulationszeichen

Eine Vielfalt an Symbolen kann über und unter den Noten erscheinen, um zu markieren, auf welche Art die Note ausgeführt werden soll. Sie werden in LilyPond notiert, indem ein Minuszeichen an die Note gehängt wird, gefolgt von dem jeweiligen Zeichen. Hier einige Beispiele:



Die Bedeutung der Zeichen kann auch verändert werden. Siehe etwa `'ly/script-init.ly'` für Beispiele.

Das Artikulationszeichen wird automatisch gesetzt, aber die Richtung kann auch erzwungen werden. Wie auch bei anderen LilyPond-Befehlen, erreicht man mit `_` eine Ausrichtung unter der Note, mit `^` eine Ausrichtung über der Note.

```
c''4^^ c''4_^
```

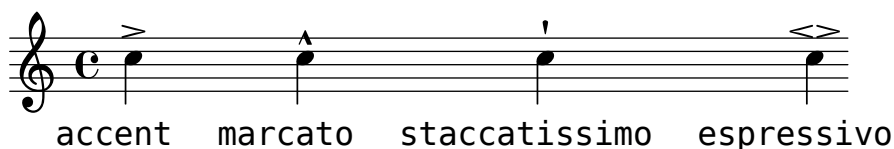


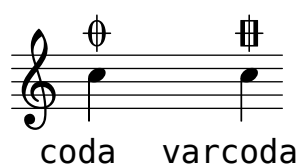
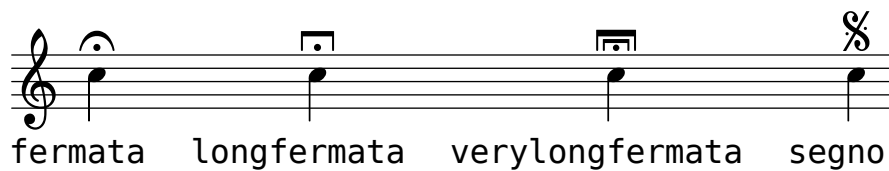
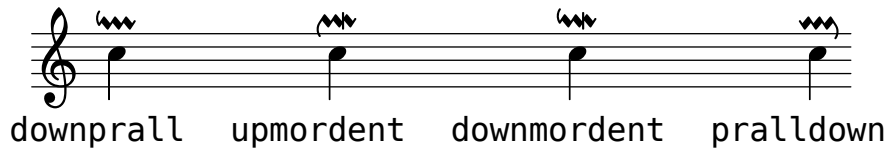
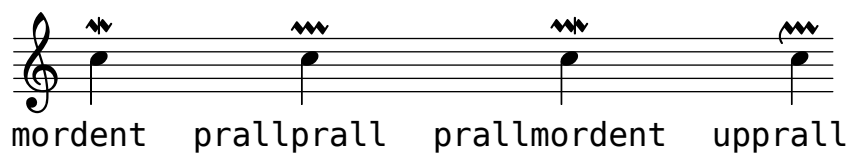
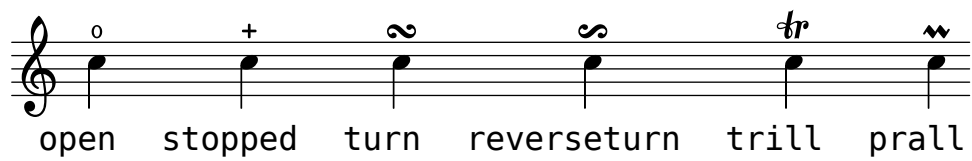
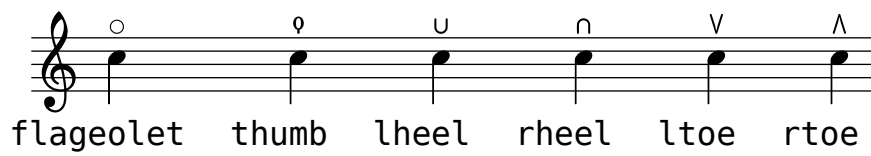
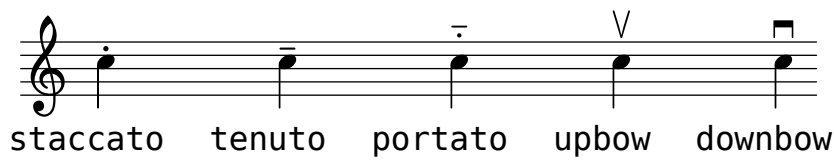
Andere Symbole können mit der Syntax `Note\Bezeichnung` hinzugefügt werden. Auch sie können mit `^` und `_` nach oben und unten gezwungen werden:

```
c\fermata c^\fermata c_ \fermata
```



Hier ist eine Liste, die alle möglichen Zeichen darstellt:





Übliche Veränderungen der Einstellungen

Die vertikale Anordnung der Zeichen wird durch die `script-priority`-Eigenschaft kontrolliert. Je kleiner die Zahl ist, umso näher wird das Zeichen an die Note gesetzt. In dem nächsten Beispiel hat das Textsymbol (`TextScript`), ein Kreuz, die niedrigste Priorität und wird also als unterstes gesetzt. Im zweiten Beispiel hat der Praller (das `Script`) die niedrigste Priorität und erscheint innen. Wenn zwei Objekte die gleiche Priorität haben, entscheidet die Reihenfolge, in der sie notiert sind, welches zuerst kommt.

```
\once \override TextScript #'script-priority = #-100
a4^\prall^\markup { \sharp }
```

```
\once \override Script #'script-priority = #-100
a4^\prall^\markup { \sharp }
```



Siehe auch

Programmreferenz: `Script`.

Fehler

Diese Zeichen erscheinen zwar im Druck, haben aber keine Auswirkung auf die produzierte MIDI-Datei.

6.6.2 Fingersatzanweisungen

Fingersatzanweisungen können folgenderweise notiert werden:

Note-Zahl

Für Fingerwechsel muss eine Textbeschriftung (`markup`) benutzt werden:

```
c4-1 c-2 c-3 c-4
c^\markup { \finger "2 - 3" }
```



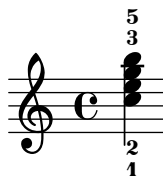
Mit dem Daumen-Befehl (`\thumb`) können die Noten bezeichnet werden, die mit dem Daumen (etwa auf dem Cello) gespielt werden sollen.

```
<a_\thumb a'-3>8 <b_\thumb b'-3>
```



Fingersätze für Akkorde können auch zu einzelnen Noten des Akkordes hinzugefügt werden, indem sie innerhalb der Akkord-Klammer direkt an die Noten angefügt werden.

```
< c-1 e-2 g-3 b-5 >4
```



Übliche Veränderungen der Einstellungen

Eine bessere Kontrolle über die Position der Fingersätze in Akkorden lässt sich mit der Eigenschaft `fingeringOrientations` herstellen:

```
\set fingeringOrientations = #'(left down)
<c-1 es-2 g-4 bes-5 > 4
\set fingeringOrientations = #'(up right down)
<c-1 es-2 g-4 bes-5 > 4
```



Mit dieser Funktion können Fingersatzbezeichnungen auch bei einstimmiger Musik sehr nah in die Notenköpfe gerückt werden.

```
\set fingeringOrientations = #'(right)
<es'-2>4
```



Siehe auch

Programmreferenz: `Fingering`.

Beispiele: `'expressive/fingering-chords.ly'`

6.6.3 Dynamik

Absolute Dynamikbezeichnung wird mit Befehlen nach den Noten angezeigt. Die vordefinierten Befehle lauten: `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz`, and `\rfz`.

```
c\ppp c\pp c \p c\mp c\mf c\f c\ff c\fff
c2\fp c\s f c\sff c\sp c\spp c\sfz c\rfz
```



Eine Crescendo-Klammer wird mit dem Befehl `\<` begonnen und mit `\!` oder einem absoluten Dynamikbefehl beendet. Ein Decrescendo beginnt mit `\>` und wird auf die gleiche Art beendet. `\cr` und `\decr` können anstelle von `\<` und `\>` benutzt werden. Weil diese Zeichen an Noten gekoppelt sind, müssen unsichtbare Noten benutzt werden, wenn mehr als ein Zeichen pro Note benötigt wird.

```
c\< c\! d\> e\!
<< f1 { s4 s4\< s4\! \> s4\! } >>
```



Eine Crescendo-Klammer beginnt normalerweise am linken Rand der Anfangsnote und endet am rechten Rand der Endnote. Wenn das Ende auf den Taktanfang fällt, endet die Klammer an der direkt vorhergehenden Taktlinie. Diese Einstellung lässt sich aber durch die Eigenschaft `hairpinToBarline` verändern.

```
\set hairpinToBarline = ##f
c4\< c2. c4\!
```



In manchen Situationen kann auch der `\espressivo`-Befehl geeignet sein, ein An- und Abschwellen einer Note anzuzeigen.

```
c2 b4 a g1\espressivo
```



Hier können allerdings sehr kurze Klammern auftreten. Der Wert von `minimum-length` in `Voice.Hairpin` kann angepasst werden, etwa:

```
\override Voice.Hairpin #'minimum-length = #5
```

Klammern können auch mit einem kleinen Kreis um die Spitze (*al niente*-Notation) gedruckt werden, wenn die `circled-tip`-Eigenschaft gesetzt wird.

```
\override Hairpin #'circled-tip = ##t
c2\< c\!
c4\> c\< c2\!
```



Anstelle der Klammern kann auch der Text *cresc.* bzw. *decr.* oder *dim.* ausgegeben werden.

```
\setTextCresc
c\< d e f\!
\setHairpinCresc
e\> d c b\!
\setTextDecresc
c\> d e f\!
\setTextDim
e\> d c b\!
```



Dieser Text kann auch beliebig angepasst werden:

```
\set crescendoText = \markup { \italic "cresc.poco" }
\set crescendoSpanner = #'dashed-line
a'2\< a a a\!\mf
```



Um neue Dynamikzeichen oder Text, der mit diesen zusammen gesetzt wird, zu erstellen, siehe den Abschnitt [Abschnitt 8.1.11 \[Neue Lautstärkezeichen\]](#), Seite 130.

Vertikale Position der Zeichen wird von der Funktion `DynamicLineSpanner` verwaltet.

Übliche Veränderungen der Einstellungen

Dynamikzeichen, die an der selben Note auftreten, werden vertikal angeordnet. Wenn Sie sicher gehen wollen, dass die Zeichen angeordnet werden, auch wenn sie nicht an der selben Note vorkommen, kann die `staff-padding`-Eigenschaft vergrößert werden.

```
\override DynamicLineSpanner #'staff-padding = #4
```

Diese Eigenschaft kann man auch benutzen, um Dynamikzeichen davor zu hindern, mit anderen Noten zusammenzustoßen.

Crescendi and Decrescendi, die an der ersten Note einer neuen Zeile enden, werden nicht ausgegeben. Mit

```
\override Score.Hairpin #'after-line-breaking = ##t
```

wird dieses Verhalten ausgeschaltet.

Text für dynamische Änderungen (wie *cresc.*) wird mit einer gestrichelten Linie gesetzt. Um diese Linie zu unterdrücken, kann der Befehl

```
\override DynamicTextSpanner #'dash-period = #-1.0
```

eingesetzt werden.

Vordefinierte Befehle

`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`.

Siehe auch

Programmreferenz: `DynamicText`, `Hairpin`. Vertikale Positionierung der Symbole wird von der Eigenschaft `DynamicLineSpanner` verwaltet.

6.6.4 Atemzeichen

Atemzeichen werden mit dem Befehl `\breathe` eingegeben.

```
c'4 \breathe d4
```



Übliche Veränderungen der Einstellungen

Das Symbol für das Atemzeichen kann verändert werden, indem die Eigenschaft `text` des `BreathingSign`-Objektes mit beliebigem Text überschrieben wird. Zum Beispiel ergibt

```
c'4
\override BreathingSign #'text
= #(make-musicglyph-markup "scripts.rvarcomma")
\breathe
d4
```



Siehe auch

Programmreferenz: `BreathingSign`.

Beispiele: `'expressive/breathing-sign.ly'`

6.6.5 Triller

Kurze Triller können wie ein normales Artikulationszeichen eingegeben werden, siehe [Abschnitt 6.6.1 \[Artikulationszeichen\]](#), Seite 101.

Längere Triller werden mit den Befehlen `\startTrillSpan` zu Beginn und `\stopTrillSpan` am Ende erstellt.

```
\new Voice {
  << { c1 \startTrillSpan }
    { s2. \grace { d16[\stopTrillSpan e] } } >>
  c4 }
```



Triller, die auf einer bestimmten Note ausgeführt werden sollen, können mit dem Befehl `pitchedTrill` notiert werden.

```
\pitchedTrill c4\startTrillSpan fis
f\stopTrillSpan
```



Dabei ist das erste Argument die Hauptnote. Die zweite Note wird ohne Hals in Klammern gesetzt.

Vordefinierte Befehle

`\startTrillSpan`, `\stopTrillSpan`.

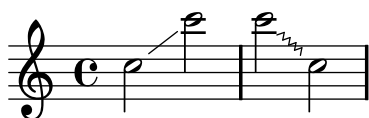
Siehe auch

Programmreferenz: `TrillSpanner`.

6.6.6 Glissando

Ein Glissando ist ein Gleiten zwischen Tonhöhen. Es wird mit einer geraden oder gezackten Linie zwischen zwei Noten notiert. Es wird mit dem Befehl `\glissando` auf eine Note folgend notiert.

```
c2\glissando c'
\override Glissando #'style = #'zigzag
c2\glissando c,
```



Siehe auch

Programmreferenz: `Glissando`.

Beispiele: `'expressive/glissando.ly'`, `'expressive/line-styles.ly'`

Fehler

Text über der Linie (wie etwa *gliss.*) wird nicht unterstützt.

6.6.7 Arpeggio

Ein Arpeggio als Zeichen, dass ein Akkord gebrochen gespielt werden soll, kann mit dem Befehl `\arpeggio` hinter dem Akkord erzeugt werden.

```
<c e g c>\arpeggio
```



Eine eckige Klammer zur Linken des Akkordes zeigt an, dass kein Arpeggio gespielt werden soll.

```
\arpeggioBracket  
<c' e g c>\arpeggio
```



Die Richtung des Arpeggios wird manchmal mit Pfeilen notiert und hat eigene Befehle.

```
\new Voice {  
  \arpeggioUp  
  <c e g c>\arpeggio  
  \arpeggioDown  
  <c e g c>\arpeggio  
}
```



Übliche Veränderungen der Einstellungen

Wenn ein Arpeggio sich über mehrere Systeme erstreckt, kann mit einem Klaviersystem die Eigenschaft `PianoStaff.connectArpeggios` gesetzt werden.

```
\new PianoStaff <<  
  \set PianoStaff.connectArpeggios = ##t  
  \new Staff { <c' e g c>\arpeggio }  
  \new Staff { \clef bass <c,, e g>\arpeggio }  
>>
```



Vordefinierte Befehle

`\arpeggio`, `\arpeggioUp`, `\arpeggioDown`, `\arpeggioNeutral`, `\arpeggioBracket`.

Siehe auch

Notationshandbuch: [Abschnitt 6.5.1 \[Bindebögen\]](#), [Seite 92](#), um Arpeggios auszuschreiben.

Programmreferenz: `Arpeggio`.

Fehler

Es ist nicht möglich, Arpeggios zwischen Systemen und solche, die sich nur auf ein System erstrecken, zum gleichen Zeitpunkt in einem Klaviersystem zu benutzen.

6.6.8 Glissando zu unbestimmter Tonhöhe

Gleiten nach oben und unten kann mit dem Befehl `\bendAfter` notiert werden.



6.7 Wiederholungszeichen

Wiederholung ist ein zentrales Konzept in der Musik, und es gibt eine ganze Vielzahl von Notationsmöglichkeiten für Wiederholungen.

6.7.1 Wiederholungstypen

Die folgenden Wiederholungsarten sind unterstützt:

- | | |
|----------------|---|
| unfold | Die wiederholte Musik wird vollständig ausgeschrieben (bzw. gespielt). Hiermit können sehr einfach sich wiederholende Stellen notiert werden. Es ist auch der einzige Wiederholungstyp, der in der MIDI-Ausgabe berücksichtigt wird. |
| volta | Wiederholungen werden nicht ausgeschrieben, aber alternative Endungen (Volta-Klammern) können bei Bedarf notiert werden. Das ist die übliche Wiederholung für Wiederholungen mit unterschiedlichen Enden. Die Wiederholung wird in der MIDI-Datei nicht berücksichtigt. |
| tremolo | Hiermit können Tremolo-Balken erstellt werden. Sie werden nicht in die MIDI-Datei aufgenommen. |
| percent | Hiermit können noten- oder taktweise Wiederholungszeichen notiert werden. Sie erinnern an das Prozentzeichen. Auch sie werden nicht in der MIDI-Datei berücksichtigt. Diese Wiederholungen müssen innerhalb eines Stimmen (Voice)-Kontextes erstellt werden. |

6.7.2 Die Syntax der Wiederholungen

LilyPond besitzt eine einzige Syntax für alle unterschiedlichen Wiederholungstypen. Sie lautet:

```
\repeat Typ Wiederholungszähler Wiederholungsnoten
```

Wenn Sie unterschiedliche Endungen haben, können Sie diese mit dem Befehl

```
\alternative {  
  Klammer1
```

```

Klammer2
Klammer3
...
}

```

wobei jede Klammer ein musikalischer Ausdruck ist. Wenn Sie nicht genug unterschiedliche Alternativen für alle Wiederholungen angeben, wird angenommen, dass die erste Alternative mehr als einmal verwendet wird.

Eine typische Wiederholung kann so aussehen:

```

c1
\repeat volta 2 { c4 d e f }
\repeat volta 2 { f e d c }

```



Und mit unterschiedlichen Klammern:

```

c1
\repeat volta 2 {c4 d e f}
\alternative { {d2 d} {f f,} }

```

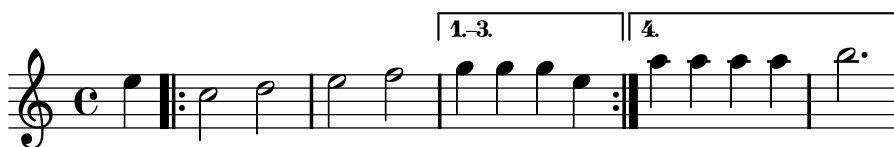


Wiederholungen können mit Auftakten kombiniert werden.

```

\new Staff {
  \partial 4 e |
  \repeat volta 4 { c2 d2 | e2 f2 | }
  \alternative { { g4 g g e } { a a a a | b2. } }
}

```

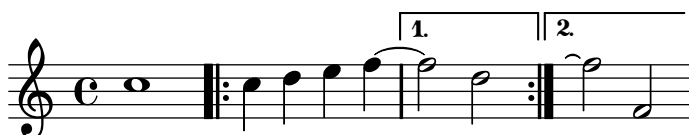


Bindebögen können auch an eine zweite Klammer angefügt werden.

```

c1
\repeat volta 2 {c4 d e f ~ }
\alternative { {f2 d} {f\repeatTie f,} }

```



Es ist auch möglich, die Klammern zu verkürzen, indem ihnen ein Wert in der Eigenschaft `voltaSpannerDuration` zugewiesen wird. Im nächsten Beispiel ist die Klammer beispielsweise nur einen 3/4-Takt lang.

```
\relative c''{
  \time 3/4
  c c c
  \set Staff.voltaSpannerDuration = #(ly:make-moment 3 4)
  \repeat "volta" 5 { d d d }
  \alternative { { e e e f f f }
    { g g g } }
}
```



Siehe auch

Beispiele:

Klammern für die Wiederholung werden normalerweise nur über dem obersten System ausgegeben. Das kann verändert werden, indem die Eigenschaft `voltaOnThisStaff` aktiviert wird. Vgl.

```
'repeats/volta-multi-staff.ly'.
```

Fehler

Eine ineinandergeschachtelte Wiederholung wie

```
\repeat ...
\repeat ...
\alternative
```

ist mehrdeutig, weil nicht klar ist, zu welchem `\repeat`-Abschnitt die `\alternative`-Endung gehört. Diese Mehrdeutigkeit wird von LilyPond aufgelöst, indem die alternative Endung immer zu der innersten Wiederholung gehört. Um Klarheit zu schaffen, bietet es sich an, in solchen Situationen Klammern zu benutzen.

Die Taktposition wird bei einer alternativen Endung nicht mitgeteilt, so dass nach einer Wiederholung diese Information manuell angegeben werden muss, entweder durch setzen von `Score.measurePosition` oder indem der Befehl `\partial` benutzt wird. Gleichmaßen werden auch Legato- oder Bindebögen nicht wiederholt.

Volta-Klammern werden nicht vertikal aneinander ausgerichtet.

6.7.3 Wiederholungen und MIDI

Mit ein bisschen Anpassung können alle Wiederholungstypen auch in der MIDI-Datei wiedergegeben werden. Das wird erreicht durch die `\unfoldRepeats`-Funktion. Hiermit werden alle Wiederholungen, welcher Art auch immer, in notengetreue Wiederholungen umgewandelt, die Noten werden also wiederholt ausgegeben.

```
\unfoldRepeats {
  \repeat tremolo 8 {c'32 e' }
```

```

\repeat percent 2 { c''8 d'' }
\repeat volta 2 {c'4 d' e' f'}
\alternative {
  { g' a' a' g' }
  {f' e' d' c' }
}
}
\bar "|"

```



Wenn man eine Partitur schreibt, die diesen `\unfoldRepeats`-Befehl für die MIDI-Ausgabe benutzt, muss man zwei `\score`-Umgebungen schreiben: eine für die MIDI-Ausgabe, wo die Wiederholungen ausgeschrieben werden, und eine für die gedruckte Notation, in der Klammern, Tremolo und Prozent-Wiederholungen benutzt werden. Zum Beispiel:

```

\score {
  ..music..
  \layout { .. }
}
\score {
  \unfoldRepeats ..music..
  \midi { .. }
}

```

6.7.4 Manuelle Wiederholungsbefehle

Die Eigenschaft `repeatCommands` kann verwendet werden, um das Aussehen der Wiederholungen zu beeinflussen. Ihr Argument ist eine Scheme-Liste an Wiederholungsbefehlen.

start-repeat

Setzt eine |: Taktlinie.

end-repeat

Setzt eine :| Taktlinie.

(volta text)

Setzt eine Volta-Klammer mit der Beschriftung *text*: Der Text kann definiert werden als Textstring oder formatierter Text, siehe Abschnitt [Abschnitt 8.1.5 \[Textbeschriftung\]](#), Seite 122. Es darf nicht vergessen werden, die Schriftart zu verändern, weil die Standardschriftart für die Nummern keine Buchstaben enthält.

(volta #f)

Beendet eine aktive Klammer.

```

c4
\set Score.repeatCommands = #'((volta "93") end-repeat)
c4 c4
\set Score.repeatCommands = #'((volta #f))
c4 c4

```



Siehe auch

Programmreferenz: `VoltaBracket`, `RepeatedMusic`, `VoltaRepeatedMusic` und `UnfoldedRepeatedMusic`.

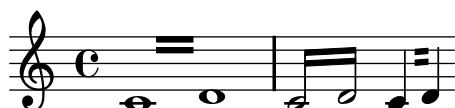
6.7.5 Tremolo-Wiederholung

Um Tremolozeichen zwischen den Noten zu setzen, kann der Wiederholungstyp `tremolo` benutzt werden.

```

\new Voice \relative c' {
  \repeat "tremolo" 8 { c16 d16 }
  \repeat "tremolo" 4 { c16 d16 }
  \repeat "tremolo" 2 { c16 d16 }
}

```



Tremolozeichen können auch einer einzelnen Noten hinzugefügt werden. In diesem Fall darf die Note nicht von Klammern eingefasst sein.

```
\repeat "tremolo" 4 c'16
```



Ähnliche Darstellung wird erreicht durch eine innere Tremolounterteilung, die im Abschnitt [Abschnitt 6.7.6 \[Tremolo-Unterteilung\]](#), Seite 115 beschrieben wird.

Siehe auch

Im Handbuch: [Abschnitt 6.7.6 \[Tremolo-Unterteilung\]](#), Seite 115, [Abschnitt 6.7 \[Wiederholungszeichen\]](#), Seite 110.

Programmreferenz: `Beam`, `StemTremolo`.

6.7.6 Tremolo-Unterteilung

Tremolozeichen können einer einzelnen Noten hinzugefügt werden, indem an sie die Zeichen `: [Anzahl]` angefügt werden. Die Anzahl bezeichnet die Dauer der einzelnen Noten, und ihr Mindestwert ist 8. Mit der Zahl 8 erhält man eine Linie durch den Notenhals. Wenn die Anzahl ausgelassen wird, wird der letzte benutzte Wert (in der Funktion `tremoloFlags` gespeichert) eingesetzt.

```
c'2:8 c':32 | c': c': |
```



Fehler

Tremolos, die auf diese Weise notiert werden, werden nicht in die MIDI-Datei aufgenommen.

Siehe auch

Im Handbuch: [Abschnitt 6.7.5 \[Tremolo-Wiederholung\]](#), Seite 114.

Programmreferenz: `StemTremolo`.

6.7.7 Taktwiederholungen

Wenn der Prozent (`percent`)-Wiederholungsstil gesetzt ist, wird eine Notenphrase wiederholt. Sie wird einmal gedruckt und dann durch ein spezielles Zeichen ersetzt. Phrasen von ein oder zwei Takten Dauer werden durch ein dem Prozentzeichen ähnlichen Zeichen markiert, Phrasen, die kürzer als ein Takt sind, durch einen Schrägstrich dargestellt. Dieser Wiederholungstyp muss innerhalb eines Stimmen (`Voice`)-Kontextes notiert werden.

```
\new Voice \relative c' {
  \repeat "percent" 4 { c4 }
  \repeat "percent" 2 { c2 es2 f4 fis4 g4 c4 }
}
```



Wiederholungen, die länger als einen Takt dauern, können gezählt werden, wenn die `countPercentRepeats`-Eigenschaft eingeschaltet wird.

```
\new Voice {
  \set countPercentRepeats = ##t
  \repeat "percent" 4 { c1 }
}
```



Isolierte Prozentzeichen können auch gedruckt werden. Das geschieht, indem einer Ganztaktpause (R) eine andere Funktion zugewiesen wird.

```
\override MultiMeasureRest #'stencil  
  = #ly:multi-measure-rest::percent  
R1
```



Siehe auch

Programmreferenz: [RepeatSlash](#), [PercentRepeat](#), [DoublePercentRepeat](#), [DoublePercentRepeatCounter](#), [PercentRepeatCounter](#), [PercentRepeatedMusic](#).

7 Instrumentenspezifische Notation

This section has not been translated yet; please refer to the manual in English.

7.1 Notation für Klavier

This section has not been translated yet; please refer to the manual in English.

7.1.1 Automatische Notensystemwechsel

This section has not been translated yet; please refer to the manual in English.

7.1.2 Manuelle Notensystemwechsel

This section has not been translated yet; please refer to the manual in English.

7.1.3 Pedalbezeichnungen

This section has not been translated yet; please refer to the manual in English.

7.1.4 Stimmführungslinien

This section has not been translated yet; please refer to the manual in English.

7.1.5 Hälse über beide Systeme

This section has not been translated yet; please refer to the manual in English.

7.2 Akkordbezeichnungen

This section has not been translated yet; please refer to the manual in English.

7.2.1 Einführung in Akkordbezeichnungen

This section has not been translated yet; please refer to the manual in English.

7.2.2 Akkord-Modus

This section has not been translated yet; please refer to the manual in English.

7.2.3 Akkordbezeichnungen drucken

This section has not been translated yet; please refer to the manual in English.

7.3 Notation von Gesang

This section has not been translated yet; please refer to the manual in English.

7.3.1 Einfache Lieder setzen

This section has not been translated yet; please refer to the manual in English.

7.3.2 Eingabe von Text

This section has not been translated yet; please refer to the manual in English.

7.3.3 Hyphens and extenders

This section has not been translated yet; please refer to the manual in English.

7.3.4 The Lyrics context

This section has not been translated yet; please refer to the manual in English.

7.3.5 Melismata

This section has not been translated yet; please refer to the manual in English.

7.3.6 Eine andere Art, den Text einzugeben

This section has not been translated yet; please refer to the manual in English.

7.3.7 Flexibilität bei der Positionierung

This section has not been translated yet; please refer to the manual in English.

7.3.7.1 Text zu mehreren Noten eines Melismas

This section has not been translated yet; please refer to the manual in English.

7.3.7.2 Getrennte Texte

This section has not been translated yet; please refer to the manual in English.

7.3.7.3 Die Melodie, die mit einer Textzeile verbunden ist, umschalten

This section has not been translated yet; please refer to the manual in English.

7.3.7.4 Specifying melismata within the lyrics

This section has not been translated yet; please refer to the manual in English.

7.3.7.5 Text unabhängig von den Noten

This section has not been translated yet; please refer to the manual in English.

7.3.8 Textabstände

This section has not been translated yet; please refer to the manual in English.

7.3.9 Mehr über Strophen

This section has not been translated yet; please refer to the manual in English.

7.3.9.1 Adding stanza numbers

7.3.9.2 Lautstärkebezeichnung hinzufügen

7.3.9.3 Sängernamen hinzufügen

7.3.9.4 Printing stanzas at the end

7.3.9.5 Printing stanzas at the end in multiple columns

7.3.10 Ambitus

This section has not been translated yet; please refer to the manual in English.

7.3.11 Weitere Vokalmusikprobleme

This section has not been translated yet; please refer to the manual in English.

7.4 Rhythmische Musik

This section has not been translated yet; please refer to the manual in English.

7.4.1 Melodierhythmus anzeigen

This section has not been translated yet; please refer to the manual in English.

7.4.2 Schlagzeugnotation

This section has not been translated yet; please refer to the manual in English.

7.4.3 Schlagzeugsysteme

This section has not been translated yet; please refer to the manual in English.

7.4.4 Geisternoten

This section has not been translated yet; please refer to the manual in English.

7.5 Gitarre

This section has not been translated yet; please refer to the manual in English.

7.5.1 Seitennummerbezeichnung

This section has not been translated yet; please refer to the manual in English.

7.5.2 Grundlagen der Tabulatur

This section has not been translated yet; please refer to the manual in English.

7.5.3 Nicht-Gitarren-Tabaturen

This section has not been translated yet; please refer to the manual in English.

7.5.4 Banjo-Tabaturen

This section has not been translated yet; please refer to the manual in English.

7.5.5 Bund-Diagramme

This section has not been translated yet; please refer to the manual in English.

7.5.6 Fingersatz der rechten Hand

This section has not been translated yet; please refer to the manual in English.

7.5.7 Weiter Gitarrenprobleme

This section has not been translated yet; please refer to the manual in English.

7.6 Dudelsack

This section has not been translated yet; please refer to the manual in English.

7.6.1 Dudelsack-Definitionen

This section has not been translated yet; please refer to the manual in English.

7.6.2 Dudelsack-Beispiele

This section has not been translated yet; please refer to the manual in English.

7.7 Notation von alter Musik

This section has not been translated yet; please refer to the manual in English.

7.7.1 Notenköpfe der alten Musik

This section has not been translated yet; please refer to the manual in English.

7.7.2 Versetzungszeichen der alten Musik

This section has not been translated yet; please refer to the manual in English.

7.7.3 Pausen der alten Musik

This section has not been translated yet; please refer to the manual in English.

7.7.4 Schlüssel der alten Musik

This section has not been translated yet; please refer to the manual in English.

7.7.5 Fähnchen der alten Musik

This section has not been translated yet; please refer to the manual in English.

7.7.6 Taktangaben der alten Musik

This section has not been translated yet; please refer to the manual in English.

7.7.7 Vorzeichen der alten Musik

This section has not been translated yet; please refer to the manual in English.

7.7.8 Custodes

This section has not been translated yet; please refer to the manual in English.

7.7.9 Divisiones

This section has not been translated yet; please refer to the manual in English.

7.7.10 Ligaturen

This section has not been translated yet; please refer to the manual in English.

7.7.10.1 Weiße Mensuralligaturen

This section has not been translated yet; please refer to the manual in English.

7.7.10.2 Ligaturen der gregorianischen Quadratnotation

This section has not been translated yet; please refer to the manual in English.

7.7.11 Gregorianische Gesangs-Kontexte

This section has not been translated yet; please refer to the manual in English.

7.7.12 Mensural-Kontexte

This section has not been translated yet; please refer to the manual in English.

7.7.13 Musica ficta-Versetzungszeichen

This section has not been translated yet; please refer to the manual in English.

7.7.14 Generalbass

This section has not been translated yet; please refer to the manual in English.

7.8 Notation für verschiedene Instrumente

This section has not been translated yet; please refer to the manual in English.

7.8.1 Flageolet (Streicher)

This section has not been translated yet; please refer to the manual in English.

8 Fortgeschrittene Notationstechniken

This section has not been translated yet; please refer to the manual in English.

8.1 Text

This section has not been translated yet; please refer to the manual in English.

8.1.1 Textarten

This section has not been translated yet; please refer to the manual in English.

8.1.2 Text und Linien

This section has not been translated yet; please refer to the manual in English.

8.1.3 Text mit Verbindungslinien

This section has not been translated yet; please refer to the manual in English.

8.1.4 Textartige Zeichen

This section has not been translated yet; please refer to the manual in English.

8.1.5 Textbeschriftung

This section has not been translated yet; please refer to the manual in English.

8.1.6 Geschachtelte Systeme

This section has not been translated yet; please refer to the manual in English.

8.1.7 Page wrapping text

This section has not been translated yet; please refer to the manual in English.

8.1.8 Überblick über Textbeschriftungsbefehle

This section has not been translated yet; please refer to the manual in English.

`\arrow-head` *axis* (integer) *direction* (direction) *filled* (boolean)

Produce an arrow head in specified direction and axis. Use the filled head if *filled* is specified.

`\beam` *width* (number) *slope* (number) *thickness* (number)

Create a beam with the specified parameters.

`\bigger` *arg* (markup)

Increase the font size relative to current setting.

`\bold` *arg* (markup)

Switch to bold font-series.

`\box` *arg* (markup)

Draw a box round *arg*. Looks at **thickness**, **box-padding** and **font-size** properties to determine line thickness and padding around the markup.

`\bracket` *arg* (markup)

Draw vertical brackets around *arg*.

`\caps` *arg* (markup)

Emit *arg* as small caps.

- `\center-align` *args* (list of markups)
Put *args* in a centered column.
- `\char` *num* (integer)
Produce a single character. For example, `\char #65` produces the letter ,A‘.
- `\circle` *arg* (markup)
Draw a circle around *arg*. Use `thickness`, `circle-padding` and `font-size` properties to determine line thickness and padding around the markup.
- `\column` *args* (list of markups)
Stack the markups in *args* vertically. The property `baseline-skip` determines the space between each markup in *args*.
- `\combine` *m1* (markup) *m2* (markup)
Print two markups on top of each other.
- `\concat` *args* (list of markups)
Concatenate *args* in a horizontal line, without spaces inbetween. Strings and simple markups are concatenated on the input level, allowing ligatures. For example, `\concat { "f" \simple #"i" }` is equivalent to "fi".
- `\dir-column` *args* (list of markups)
Make a column of *args*, going up or down, depending on the setting of the `#'direction` layout property.
- `\doubleflat`
Draw a double flat symbol.
- `\doublessharp`
Draw a double sharp symbol.
- `\draw-circle` *radius* (number) *thickness* (number) *fill* (boolean)
A circle of radius *radius*, thickness *thickness* and optionally filled.
- `\draw-line` *dest* (pair of numbers)
A simple line. Uses the `thickness` property.
- `\dynamic` *arg* (markup)
Use the dynamic font. This font only contains **s**, **f**, **m**, **z**, **p**, and **r**. When producing phrases, like ,più **f**, the normal words (like ,più‘) should be done in a different font. The recommended font for this is bold and italic.
- `\epsfile` *axis* (number) *size* (number) *file-name* (string)
Inline an EPS image. The image is scaled along *axis* to *size*.
- `\fill-line` *markups* (list of markups)
Put *markups* in a horizontal line of width *line-width*. The markups are spaced or flushed to fill the entire line. If there are no arguments, return an empty stencil.
- `\filled-box` *xext* (pair of numbers) *yext* (pair of numbers) *blot* (number)
Draw a box with rounded corners of dimensions *xext* and *yext*. For example,
`\filled-box #'(-.3 . 1.8) #'(-.3 . 1.8) #0`
creates a box extending horizontally from -0.3 to 1.8 and vertically from -0.3 up to 1.8, with corners formed from a circle of diameter 0 (i.e. sharp corners).
- `\finger` *arg* (markup)
Set the argument as small numbers.
- `\flat`
Draw a flat symbol.

`\fontCaps` *arg* (markup)

Set `font-shape` to `caps`.

`\fontsize` *increment* (number) *arg* (markup)

Add *increment* to the font-size. Adjust baseline skip accordingly.

`\fraction` *arg1* (markup) *arg2* (markup)

Make a fraction of two markups.

`\fret-diagram` *definition-string* (string)

Make a (guitar) fret diagram. For example, say

```
\markup \fret-diagram #"s:0.75;6-x;5-x;4-o;3-2;2-3;1-2;"
```

for fret spacing 3/4 of staff space, D chord diagram

Syntax rules for *definition-string*:

- Diagram items are separated by semicolons.
- Possible items:
 - `s:number` – Set the fret spacing of the diagram (in staff spaces). Default: 1.
 - `t:number` – Set the line thickness (in staff spaces). Default: 0.05.
 - `h:number` – Set the height of the diagram in frets. Default: 4.
 - `w:number` – Set the width of the diagram in strings. Default: 6.
 - `f:number` – Set fingering label type (0 = none, 1 = in circle on string, 2 = below string). Default: 0.
 - `d:number` – Set radius of dot, in terms of fret spacing. Default: 0.25.
 - `p:number` – Set the position of the dot in the fret space. 0.5 is centered; 1 is on lower fret bar, 0 is on upper fret bar. Default: 0.6.
 - `c:string1-string2-fret` – Include a barre mark from *string1* to *string2* on *fret*.
 - `string-fret` – Place a dot on *string* at *fret*. If *fret* is ‘o’, *string* is identified as open. If *fret* is ‘x’, *string* is identified as muted.
 - `string-fret-fingering` – Place a dot on *string* at *fret*, and label with *fingering* as defined by the `f:` code.
- Note: There is no limit to the number of fret indications per string.

`\fret-diagram-terse` *definition-string* (string)

Make a fret diagram markup using terse string-based syntax.

Here an example

```
\markup \fret-diagram-terse #"x;x;o;2;3;2;"
```

for a D chord diagram.

Syntax rules for *definition-string*:

- Strings are terminated by semicolons; the number of semicolons is the number of strings in the diagram.
- Mute strings are indicated by ‘x’.
- Open strings are indicated by ‘o’.
- A number indicates a fret indication at that fret.
- If there are multiple fret indicators desired on a string, they should be separated by spaces.
- Fingerings are given by following the fret number with a -, followed by the finger indicator, e.g. ‘3-2’ for playing the third fret with the second finger.

- Where a barre indicator is desired, follow the fret (or fingering) symbol with `-(` to start a barre and `-)` to end the barre.

`\fret-diagram-verbose` *marking-list* (list)

Make a fret diagram containing the symbols indicated in *marking-list*.

For example,

```
\markup \fret-diagram-verbose
#'( (mute 6) (mute 5) (open 4)
    (place-fret 3 2) (place-fret 2 3) (place-fret 1 2) )
```

produces a standard D chord diagram without fingering indications.

Possible elements in *marking-list*:

`(mute string-number)`

Place a small ,x‘ at the top of string *string-number*.

`(open string-number)`

Place a small ,o‘ at the top of string *string-number*.

`(barre start-string end-string fret-number)`

Place a barre indicator (much like a tie) from string *start-string* to string *end-string* at fret *fret-number*.

`(place-fret string-number fret-number finger-value)`

Place a fret playing indication on string *string-number* at fret *fret-number* with an optional fingering label *finger-value*. By default, the fret playing indicator is a solid dot. This can be changed by setting the value of the variable *dot-color*. If the *finger* part of the `place-fret` element is present, *finger-value* will be displayed according to the setting of the variable *finger-code*. There is no limit to the number of fret indications per string.

`\fromproperty` *symbol* (symbol)

Read the *symbol* from property settings, and produce a stencil from the markup contained within. If *symbol* is not defined, it returns an empty markup.

`\general-align` *axis* (integer) *dir* (number) *arg* (markup)

Align *arg* in *axis* direction to the *dir* side.

`\halign` *dir* (number) *arg* (markup)

Set horizontal alignment. If *dir* is `-1`, then it is left-aligned, while `+1` is right. Values inbetween interpolate alignment accordingly.

`\hbracket` *arg* (markup)

Draw horizontal brackets around *arg*.

`\hcenter-in` *length* (number) *arg* (markup)

Center *arg* horizontally within a box of extending *length*/2 to the left and right.

`\hcenter` *arg* (markup)

Align *arg* to its X center.

`\hspace` *amount* (number)

This produces an invisible object taking horizontal space. For example,

```
\markup { A \hspace #2.0 B }
```

puts extra space between A and B, on top of the space that is normally inserted before elements on a line.

`\huge` *arg* (markup)

Set font size to `+2`.

- `\italic arg` (markup)
Use italic **font-shape** for *arg*.
- `\justify-field symbol` (symbol)
Justify the data which has been assigned to *symbol*.
- `\justify args` (list of markups)
Like wordwrap, but with lines stretched to justify the margins. Use `\override #'(line-width . X)` to set the line width; *X* is the number of staff spaces.
- `\justify-string arg` (string)
Justify a string. Paragraphs may be separated with double newlines
- `\large arg` (markup)
Set font size to +1.
- `\larger arg` (markup)
Copy of the bigger-markup command.
- `\left-align arg` (markup)
Align *arg* on its left edge.
- `\line args` (list of markups)
Put *args* in a horizontal line. The property **word-space** determines the space between each markup in *args*.
- `\lookup glyph-name` (string)
Lookup a glyph by name.
- `\lower amount` (number) *arg* (markup)
Lower *arg* by the distance *amount*. A negative *amount* indicates raising; see also `\raise`.
- `\magnify sz` (number) *arg* (markup)
Set the font magnification for its argument. In the following example, the middle A is 10% larger:

$$A \text{ \magnify \#1.1 } \{ A \} A$$
Note: Magnification only works if a font name is explicitly selected. Use `\fontsize` otherwise.
- `\markalphabet num` (integer)
Make a markup letter for *num*. The letters start with A to Z and continue with double letters.
- `\markletter num` (integer)
Make a markup letter for *num*. The letters start with A to Z (skipping letter I), and continue with double letters.
- `\medium arg` (markup)
Switch to medium font series (in contrast to bold).
- `\musicglyph glyph-name` (string)
glyph-name is converted to a musical symbol; for example, `\musicglyph \#"accidentals.natural"` selects the natural sign from the music font. See user manual, **The Feta font** for a complete listing of the possible glyphs.
- `\natural`
Draw a natural symbol.
- `\normal-size-sub arg` (markup)
Set *arg* in subscript, in a normal font size.

`\normal-size-super arg` (markup)

Set *arg* in superscript with a normal font size.

`\normal-text arg` (markup)

Set all font related properties (except the size) to get the default normal text font, no matter what font was used earlier.

`\normalsize arg` (markup)

Set font size to default.

`\note-by-number log` (number) *dot-count* (number) *dir* (number)

Construct a note symbol, with stem. By using fractional values for *dir*, you can obtain longer or shorter stems.

`\note duration` (string) *dir* (number)

This produces a note with a stem pointing in *dir* direction, with the *duration* for the note head type and augmentation dots. For example, `\note #"4." #-0.75` creates a dotted quarter note, with a shortened down stem.

`\null`

An empty markup with extents of a single point.

`\number arg` (markup)

Set font family to **number**, which yields the font used for time signatures and fingerings. This font only contains numbers and some punctuation. It doesn't have any letters.

`\on-the-fly procedure` (symbol) *arg* (markup)

Apply the *procedure* markup command to *arg*. *procedure* should take a single argument.

`\override new-prop` (pair) *arg* (markup)

Add the first argument in to the property list. Properties may be any sort of property supported by **font-interface** and **text-interface**, for example

```
\override #'(font-family . married) "bla"
```

`\pad-around amount` (number) *arg* (markup)

Add padding *amount* all around *arg*.

`\pad-markup padding` (number) *arg* (markup)

Add space around a markup object.

`\pad-to-box x-ext` (pair of numbers) *y-ext* (pair of numbers) *arg* (markup)

Make *arg* take at least *x-ext*, *y-ext* space.

`\pad-x amount` (number) *arg* (markup)

Add padding *amount* around *arg* in the X direction.

`\page-ref label` (symbol) *gauge* (markup) *default* (markup)

Reference to a page number. *label* is the label set on the referenced page (using the `\label` command), *gauge* a markup used to estimate the maximum width of the page number, and *default* the value to display when *label* is not found.

`\postscript str` (string)

This inserts *str* directly into the output as a PostScript command string. Due to technicalities of the output backends, different scales should be used for the T_EX and PostScript backend, selected with `-f`.

For the T_EX backend, the following string prints a rotated text

```
0 0 moveto /ecrm10 findfont
1.75 scalefont setfont 90 rotate (hello) show
```

The magical constant 1.75 scales from LilyPond units (staff spaces) to T_EX dimensions.

For the postscript backend, use the following

```
gsave /ecrm10 findfont
10.0 output-scale div
scalefont setfont 90 rotate (hello) show grestore
```

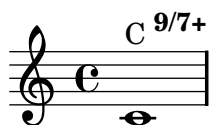
`\put-adjacent arg1 (markup) axis (integer) dir (direction) arg2 (markup)`

Put *arg2* next to *arg1*, without moving *arg1*.

`\raise amount (number) arg (markup)`

Raise *arg* by the distance *amount*. A negative *amount* indicates lowering, see also `\lower`.

```
c1^\markup { C \small \raise #1.0 \bold { "9/7+" } }
```



The argument to `\raise` is the vertical displacement amount, measured in (global) staff spaces. `\raise` and `\super` raise objects in relation to their surrounding markups.

If the text object itself is positioned above or below the staff, then `\raise` cannot be used to move it, since the mechanism that positions it next to the staff cancels any shift made with `\raise`. For vertical positioning, use the `padding` and/or `extra-offset` properties.

`\right-align arg (markup)`

Align *arg* on its right edge.

`\roman arg (markup)`

Set font family to `roman`.

`\rotate ang (number) arg (markup)`

Rotate object with *ang* degrees around its center.

`\sans arg (markup)`

Switch to the sans serif family.

`\score score (unknown)`

Inline an image of music.

`\semiflat`

Draw a semiflat.

`\semisharp`

Draw a semi sharp symbol.

`\sesquiflat`

Draw a 3/2 flat symbol.

`\sesquisharp`

Draw a 3/2 sharp symbol.

`\sharp`

Draw a sharp symbol.

`\simple str` (string)

A simple text string; `\markup { foo }` is equivalent with `\markup { \simple #"foo" }`.

`\slashed-digit num` (integer)

A feta number, with slash. This is for use in the context of figured bass notation.

`\small arg` (markup)

Set font size to -1.

`\smallCaps text` (markup)

Turn `text`, which should be a string, to small caps.

`\markup \smallCaps "Text between double quotes"`

`\smaller arg` (markup)

Decrease the font size relative to current setting.

`\stencil stil` (unknown)

Use a stencil as markup.

`\strut`

Create a box of the same height as the space in the current font.

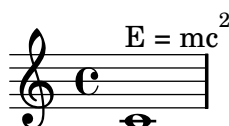
`\sub arg` (markup)

Set `arg` in subscript.

`\super arg` (markup)

Raising and lowering texts can be done with `\super` and `\sub`:

`c1~\markup { E "=" \concat { "mc" \super "2" } }`



`\teeny arg` (markup)

Set font size to -3.

`\text arg` (markup)

Use a text font instead of music symbol or music alphabet font.

`\tied-lyric str` (string)

Like simple-markup, but use tie characters for ,~‘ tilde symbols.

`\tiny arg` (markup)

Set font size to -2.

`\translate offset` (pair of numbers) `arg` (markup)

This translates an object. Its first argument is a cons of numbers.

`A \translate #(cons 2 -3) { B C } D`

This moves ,B C‘ 2 spaces to the right, and 3 down, relative to its surroundings. This command cannot be used to move isolated scripts vertically, for the same reason that `\raise` cannot be used for that.

`\translate-scaled offset` (pair of numbers) `arg` (markup)

Translate `arg` by `offset`, scaling the offset by the `font-size`.

- `\transparent` *arg* (markup)
Make the argument transparent.
- `\triangle` *filled* (boolean)
A triangle, either filled or empty.
- `\typewriter` *arg* (markup)
Use `font-family typewriter` for *arg*.
- `\upright` *arg* (markup)
Set font shape to `upright`. This is the opposite of `italic`.
- `\vcenter` *arg* (markup)
Align *arg* to its Y center.
- `\verbatim-file` *name* (string)
Read the contents of a file, and include it verbatim.
- `\whiteout` *arg* (markup)
Provide a white underground for *arg*.
- `\with-color` *color* (list) *arg* (markup)
Draw *arg* in color specified by *color*.
- `\with-dimensions` *x* (pair of numbers) *y* (pair of numbers) *arg* (markup)
Set the dimensions of *arg* to *x* and *y*.
- `\with-url` *url* (string) *arg* (markup)
Add a link to URL *url* around *arg*. This only works in the PDF backend.
- `\wordwrap-field` *symbol* (symbol)
Wordwrap the data which has been assigned to *symbol*.
- `\wordwrap` *args* (list of markups)
Simple wordwrap. Use `\override #'(line-width . X)` to set the line width, where *X* is the number of staff spaces.
- `\wordwrap-string` *arg* (string)
Wordwrap a string. Paragraphs may be separated with double newlines.

8.1.9 Overview of text markup list commands

This section has not been translated yet; please refer to the manual in English.

- `\column-lines` *args* (list of markups)
Like `\column`, but return a list of lines instead of a single markup. `baseline-skip` determines the space between each markup in *args*.
- `\justified-lines` *args* (list of markups)
Like `\justify`, but return a list of lines instead of a single markup. Use `\override #'(line-width . X)` to set the line width; *X* is the number of staff spaces.
- `\override-lines` *new-prop* (pair) *args* (list of markups)
Like `\override`, for markup lists.
- `\wordwrap-lines` *args* (list of markups)
Like `\wordwrap`, but return a list of lines instead of a single markup. Use `\override #'(line-width . X)` to set the line width, where *X* is the number of staff spaces.

8.1.10 Auswahl der Schriftart

This section has not been translated yet; please refer to the manual in English.

8.1.11 Neue Lautstärkezeichen

This section has not been translated yet; please refer to the manual in English.

8.2 Orchesterstimmen vorbereiten

This section has not been translated yet; please refer to the manual in English.

8.2.1 Mehrtaktige Pausen

This section has not been translated yet; please refer to the manual in English.

8.2.2 Metronombezeichnung

This section has not been translated yet; please refer to the manual in English.

8.2.3 Übungszeichen

This section has not been translated yet; please refer to the manual in English.

8.2.4 Taktnummern

This section has not been translated yet; please refer to the manual in English.

8.2.5 Instrumentenbezeichnungen

This section has not been translated yet; please refer to the manual in English.

8.2.6 Transposition von Instrumenten

This section has not been translated yet; please refer to the manual in English.

8.2.7 Oktavierungsklammern

This section has not been translated yet; please refer to the manual in English.

8.2.8 Verschiedene Editionen aus einer Quelldatei

This section has not been translated yet; please refer to the manual in English.

8.3 Orchestermusik

This section has not been translated yet; please refer to the manual in English.

8.3.1 Automatische Kombination von Stimmen

This section has not been translated yet; please refer to the manual in English.

8.3.2 Systeme verstecken

This section has not been translated yet; please refer to the manual in English.

8.3.3 Stichnoten

This section has not been translated yet; please refer to the manual in English.

8.3.4 Stichnoten formatieren

This section has not been translated yet; please refer to the manual in English.

8.3.5 An Kadenzen ausrichten

This section has not been translated yet; please refer to the manual in English.

8.4 Moderne Notation

This section has not been translated yet; please refer to the manual in English.

8.4.1 Polymetrische Notation

This section has not been translated yet; please refer to the manual in English.

8.4.2 Verwaltung der Zeiteinheiten

This section has not been translated yet; please refer to the manual in English.

8.4.3 Proportionale Notation (Einleitung)

This section has not been translated yet; please refer to the manual in English.

8.4.4 Cluster

This section has not been translated yet; please refer to the manual in English.

8.4.5 Besondere Notenköpfe

This section has not been translated yet; please refer to the manual in English.

8.4.6 Gespreizte Balken

This section has not been translated yet; please refer to the manual in English.

8.4.7 Improvisation

This section has not been translated yet; please refer to the manual in English.

8.4.8 Auswahl der Notations-Schriftgröße

This section has not been translated yet; please refer to the manual in English.

8.5 Pädagogische Verwendung

This section has not been translated yet; please refer to the manual in English.

8.5.1 Erklärungsblasen

This section has not been translated yet; please refer to the manual in English.

8.5.2 Ein leeres Notenblatt

This section has not been translated yet; please refer to the manual in English.

8.5.3 Unsichtbare Noten

This section has not been translated yet; please refer to the manual in English.

8.5.4 Notenköpfe mit besonderen Formen

This section has not been translated yet; please refer to the manual in English.

8.5.5 Easy Notation-Notenköpfe

This section has not been translated yet; please refer to the manual in English.

8.5.6 Analyseklammern

This section has not been translated yet; please refer to the manual in English.

8.5.7 Farbige Objekte

This section has not been translated yet; please refer to the manual in English.

8.5.8 Klammern

This section has not been translated yet; please refer to the manual in English.

8.5.9 Gitternetzlinien

This section has not been translated yet; please refer to the manual in English.

9 Standardeinstellungen verändern

This section has not been translated yet; please refer to the manual in English.

9.1 Automatic notation

This section has not been translated yet; please refer to the manual in English.

9.1.1 Automatische Versetzungszeichen

This section has not been translated yet; please refer to the manual in English.

9.1.2 Einstellung von automatischen Balken

This section has not been translated yet; please refer to the manual in English.

9.2 Interpretationsumgebungen

This section has not been translated yet; please refer to the manual in English.

9.2.1 Was sind Umgebungen?

This section has not been translated yet; please refer to the manual in English.

9.2.2 Umgebungen erstellen

This section has not been translated yet; please refer to the manual in English.

9.2.3 Umgebungseigenschaften lokal ändern

This section has not been translated yet; please refer to the manual in English.

9.2.4 Umgebungs-Plugins verändern

This section has not been translated yet; please refer to the manual in English.

9.2.5 Layouteinstellungen mit Umgebungen

This section has not been translated yet; please refer to the manual in English.

9.2.6 Die Standardeinstellungen von Umgebungen ändern

This section has not been translated yet; please refer to the manual in English.

9.2.7 Neue Umgebungen definieren

This section has not been translated yet; please refer to the manual in English.

9.2.8 Umgebungen aneinander ausrichten

This section has not been translated yet; please refer to the manual in English.

9.2.9 Vertical grouping of grobs

This section has not been translated yet; please refer to the manual in English.

9.3 The `\override` command

This section has not been translated yet; please refer to the manual in English.

9.3.1 Eine Korrektur konstruieren

This section has not been translated yet; please refer to the manual in English.

9.3.2 Zurechtfinden in der Programmreferenz

This section has not been translated yet; please refer to the manual in English.

9.3.3 Layout-Schnittstellen

This section has not been translated yet; please refer to the manual in English.

9.3.4 Die Grob-Eigenschaften

This section has not been translated yet; please refer to the manual in English.

9.3.5 Objekte, die mit der Eingabe verbunden sind

This section has not been translated yet; please refer to the manual in English.

9.3.6 Using Scheme code instead of `\tweak`

This section has not been translated yet; please refer to the manual in English.

9.3.7 `\set` vs. `\override`

This section has not been translated yet; please refer to the manual in English.

9.3.8 Schwierige Korrekturen

This section has not been translated yet; please refer to the manual in English.

10 Nichtmusikalische Notation

This section has not been translated yet; please refer to the manual in English.

10.1 Quelldateien

This section has not been translated yet; please refer to the manual in English.

10.1.1 Die Dateistruktru (Einleitung)

This section has not been translated yet; please refer to the manual in English.

10.1.2 Die Dateistruktur

This section has not been translated yet; please refer to the manual in English.

10.1.3 Ein einzelner musikalischer Ausdruck

This section has not been translated yet; please refer to the manual in English.

10.1.4 Mehrere Partituren in einem Buch

This section has not been translated yet; please refer to the manual in English.

10.1.5 Fragmente der Notation extrahieren

This section has not been translated yet; please refer to the manual in English.

10.1.6 LilyPond-Dateien einfügen

This section has not been translated yet; please refer to the manual in English.

10.1.7 Textkodierung

This section has not been translated yet; please refer to the manual in English.

10.2 Titel

This section has not been translated yet; please refer to the manual in English.

10.2.1 Titel erstellen

This section has not been translated yet; please refer to the manual in English.

10.2.2 Eigene Titel

This section has not been translated yet; please refer to the manual in English.

10.2.3 Reference to page numbers

This section has not been translated yet; please refer to the manual in English.

10.2.4 Table of contents

This section has not been translated yet; please refer to the manual in English.

10.3 MDID-Ausgabe

This section has not been translated yet; please refer to the manual in English.

10.3.1 MIDI-Dateien erstellen

This section has not been translated yet; please refer to the manual in English.

10.3.2 Der MIDI-Block

This section has not been translated yet; please refer to the manual in English.

10.3.3 MIDI-Instrumentenbezeichnungen

This section has not been translated yet; please refer to the manual in English.

10.4 LilyPond-Notation anzeigen

This section has not been translated yet; please refer to the manual in English.

10.5 Korrigierte Musik überspringen

This section has not been translated yet; please refer to the manual in English.

11 Abstände

This section has not been translated yet; please refer to the manual in English.

11.1 Papier und Seiten

This section has not been translated yet; please refer to the manual in English.

11.1.1 Papiergröße

This section has not been translated yet; please refer to the manual in English.

11.1.2 Seitenformatierung

This section has not been translated yet; please refer to the manual in English.

11.2 Notenlayout

This section has not been translated yet; please refer to the manual in English.

11.2.1 Die Notensystemgröße einstellen

This section has not been translated yet; please refer to the manual in English.

11.2.2 Partiturlayout

This section has not been translated yet; please refer to the manual in English.

11.3 Abstände anzeigen lassen

This section has not been translated yet; please refer to the manual in English.

11.4 Umbrüche

This section has not been translated yet; please refer to the manual in English.

11.4.1 Zeilenumbrüche

This section has not been translated yet; please refer to the manual in English.

11.4.2 Seitenumbrüche

This section has not been translated yet; please refer to the manual in English.

11.4.3 Optimale Seitenumbrüche

This section has not been translated yet; please refer to the manual in English.

11.4.4 Optimale Umbrüche zum Blättern

This section has not been translated yet; please refer to the manual in English.

11.4.5 Ausdrückliche Umbrüche

This section has not been translated yet; please refer to the manual in English.

11.4.6 Eine zusätzliche Stimme für Umbrüche benutzen

This section has not been translated yet; please refer to the manual in English.

11.5 Vertikale Abstände

This section has not been translated yet; please refer to the manual in English.

11.5.1 Vertikale Abstände innerhalb eines Systemes

This section has not been translated yet; please refer to the manual in English.

11.5.2 Vertikale Abstände zwischen Systemen

This section has not been translated yet; please refer to the manual in English.

11.5.3 Explizite Positionierung von Systemen

This section has not been translated yet; please refer to the manual in English.

11.5.4 Vertikale Abstände mit zwei Durchgängen

This section has not been translated yet; please refer to the manual in English.

11.5.5 Vermeidung von vertikalen Zusammenstößen

This section has not been translated yet; please refer to the manual in English.

11.6 Horizontale Abstände

This section has not been translated yet; please refer to the manual in English.

11.6.1 Überblick über horizontale Abstände

This section has not been translated yet; please refer to the manual in English.

11.6.2 Eine neue Umgebung mit anderen Abständen

This section has not been translated yet; please refer to the manual in English.

11.6.3 Horizontale Abstände verändern

This section has not been translated yet; please refer to the manual in English.

11.6.4 Zeilenlänge

This section has not been translated yet; please refer to the manual in English.

11.6.5 Proportionale Notation

This section has not been translated yet; please refer to the manual in English.

12 Schnittstellen für Programmierer

This section has not been translated yet; please refer to the manual in English.

12.1 Musikalische Funktionen

This section has not been translated yet; please refer to the manual in English.

12.1.1 Überblick über musikalische Funktionen

This section has not been translated yet; please refer to the manual in English.

12.1.2 Einfache Ersetzungsfunktionen

This section has not been translated yet; please refer to the manual in English.

12.1.3 Paarige Ersetzungsfunktionen

This section has not been translated yet; please refer to the manual in English.

12.1.4 Mathematik in Funktionen

This section has not been translated yet; please refer to the manual in English.

12.1.5 Leere Funktionen

This section has not been translated yet; please refer to the manual in English.

12.1.6 Funktionen ohne Argumente

This section has not been translated yet; please refer to the manual in English.

12.1.7 Überblick über vorhandene musikalische Funktionen

This section has not been translated yet; please refer to the manual in English.

`transposedCueDuring` - *what* (string) *dir* (direction) *pitch-note* (music) *main-music* (music)
(undocumented; fixme)

`displayLilyMusic` - *music* (music)
(undocumented; fixme)

`tocItem` - *text* (markup)
Add a line to the table of content, using the `tocItemMarkup` paper variable markup

`appoggiatura` - *music* (music)
(undocumented; fixme)

`cueDuring` - *what* (string) *dir* (direction) *main-music* (music)
(undocumented; fixme)

`removeWithTag` - *tag* (symbol) *music* (music)
(undocumented; fixme)

`breathe` -
(undocumented; fixme)

`clef` - *type* (string)
(undocumented; fixme)

`overrideProperty` - *name* (string) *property* (symbol) *value* (any type)
(undocumented; fixme)

withMusicProperty - *sym* (symbol) *val* (any type) *music* (music)
 (undocumented; fixme)

assertBeamSlope - *comp* (procedure)
 (undocumented; fixme)

acciaccatura - *music* (music)
 (undocumented; fixme)

pitchedTrill - *main-note* (music) *secondary-note* (music)
 (undocumented; fixme)

applyContext - *proc* (procedure)
 (undocumented; fixme)

allowPageTurn -
 (undocumented; fixme)

assertBeamQuant - *l* (pair) *r* (pair)
 (undocumented; fixme)

includePageLayoutFile -
 (undocumented; fixme)

transposition - *pitch-note* (music)
 (undocumented; fixme)

applyOutput - *ctx* (symbol) *proc* (procedure)
 (undocumented; fixme)

afterGrace - *main* (music) *grace* (music)
 (undocumented; fixme)

label - *label* (symbol)
 (undocumented; fixme)

oldaddyrics - *music* (music) *lyrics* (music)
 (undocumented; fixme)

pageTurn -
 (undocumented; fixme)

compressMusic - *fraction* (pair of numbers) *music* (music)
 (undocumented; fixme)

featherDurations - *factor* (moment) *argument* (music)
 (undocumented; fixme)

displayMusic - *music* (music)
 (undocumented; fixme)

parallelMusic - *voice-ids* (list) *music* (music)
 (undocumented; fixme)

resetRelativeOctave - *reference-note* (music)
 (undocumented; fixme)

addQuote - *name* (string) *music* (music)
 (undocumented; fixme)

octave - *pitch-note* (music)
 (undocumented; fixme)

`parenthesize` - *arg* (music)
(undocumented; fixme)

`balloonGrobText` - *grob-name* (symbol) *offset* (pair of numbers) *text* (markup)
(undocumented; fixme)

`instrumentSwitch` - *name* (string)
(undocumented; fixme)

`makeClusters` - *arg* (music)
(undocumented; fixme)

`spacingTweaks` - *parameters* (list)
(undocumented; fixme)

`tag` - *tag* (symbol) *arg* (music)
(undocumented; fixme)

`noPageTurn` -
(undocumented; fixme)

`bendAfter` - *delta* (integer)
(undocumented; fixme)

`partcombine` - *part1* (music) *part2* (music)
(undocumented; fixme)

`grace` - *music* (music)
(undocumented; fixme)

`noPageBreak` -
(undocumented; fixme)

`pageBreak` -
(undocumented; fixme)

`bar` - *type* (string)
(undocumented; fixme)

`shiftDurations` - *dur* (integer) *dots* (integer) *arg* (music)
(undocumented; fixme)

`unfoldRepeats` - *music* (music)
(undocumented; fixme)

`balloonText` - *offset* (pair of numbers) *text* (markup)
(undocumented; fixme)

`quoteDuring` - *what* (string) *main-music* (music)
(undocumented; fixme)

`barNumberCheck` - *n* (integer)
(undocumented; fixme)

`addInstrumentDefinition` - *name* (string) *lst* (list)
(undocumented; fixme)

`scoreTweak` - *name* (string)
(undocumented; fixme)

`autochange` - *music* (music)
(undocumented; fixme)

`rightHandFinger` - *finger* (number or string)
 (undocumented; fixme)

`endSpanners` - *music* (music)
 (undocumented; fixme)

`musicMap` - *proc* (procedure) *mus* (music)
 (undocumented; fixme)

`applyMusic` - *func* (procedure) *music* (music)
 (undocumented; fixme)

`killCues` - *music* (music)
 (undocumented; fixme)

`keepWithTag` - *tag* (symbol) *music* (music)
 (undocumented; fixme)

`tweak` - *sym* (symbol) *val* (any type) *arg* (music)
 (undocumented; fixme)

12.2 Schnittstelle für Programmierer

This section has not been translated yet; please refer to the manual in English.

12.2.1 Eingabevariablen und Scheme

This section has not been translated yet; please refer to the manual in English.

12.2.2 Interne Repräsentation der Musik

This section has not been translated yet; please refer to the manual in English.

12.3 Komplizierte Funktionen erstellen

This section has not been translated yet; please refer to the manual in English.

12.3.1 Musikalische Funktionen darstellen

This section has not been translated yet; please refer to the manual in English.

12.3.2 Eigenschaften von Musikobjekten

This section has not been translated yet; please refer to the manual in English.

12.3.3 Verdopplung einer Note mit Bindebögen (Beispiel)

This section has not been translated yet; please refer to the manual in English.

12.3.4 Artikulationszeichen zu Noten hinzufügen (Beispiel)

This section has not been translated yet; please refer to the manual in English.

12.4 Programmierungsschnittstelle für Textbeschriftungen

This section has not been translated yet; please refer to the manual in English.

12.4.1 Beschriftungskonstruktionen in Scheme

This section has not been translated yet; please refer to the manual in English.

12.4.2 Wie Beschriftungen intern funktionieren

This section has not been translated yet; please refer to the manual in English.

12.4.3 Neue Definitionen von Beschriftungsbefehlen

This section has not been translated yet; please refer to the manual in English.

12.4.4 New markup list command definition

This section has not been translated yet; please refer to the manual in English.

12.5 Kontexte für Programmierer

This section has not been translated yet; please refer to the manual in English.

12.5.1 Kontextauswertung

This section has not been translated yet; please refer to the manual in English.

12.5.2 Eine Funktion auf alle Layout-Objekte anwenden

This section has not been translated yet; please refer to the manual in English.

12.6 Scheme-Vorgänge als Eigenschaften

This section has not been translated yet; please refer to the manual in English.

Anhang A Literatur

This section has not been translated yet; please refer to the manual in English.

Anhang B Scheme-Übung

This section has not been translated yet; please refer to the manual in English.

Anhang C Notationsübersicht

This section has not been translated yet; please refer to the manual in English.

C.1 Liste der Akkordbezeichnungen

This section has not been translated yet; please refer to the manual in English.

C.2 MIDI-Instrumente

This section has not been translated yet; please refer to the manual in English.

C.3 Liste der Farben

This section has not been translated yet; please refer to the manual in English.

Normale Farben

X-Farbbezeichnungen

Farben ohne eine numerale Endung

Farben mit einer numeralen Endung

Grauskala

C.4 Die Feta-Schriftart

This section has not been translated yet; please refer to the manual in English.

C.5 Notenkopfstile

This section has not been translated yet; please refer to the manual in English.

Anhang D Vorlagen

This section has not been translated yet; please refer to the manual in English.

D.1 Ein einzelnes System

This section has not been translated yet; please refer to the manual in English.

D.1.1 Nur Noten

D.1.2 Noten und Text

D.1.3 Noten und Akkordbezeichnungen

D.1.4 Noten, Text und Akkordbezeichnungen

D.2 Klaviervorlagen

This section has not been translated yet; please refer to the manual in English.

D.2.1 Piano Solo

D.2.2 Klavier und Gesangstimme

D.2.3 Klavier mit zentriertem Text

D.2.4 Klavier mit zentrierten Lautstärkebezeichnungen

D.3 Streichquartett

This section has not been translated yet; please refer to the manual in English.

D.3.1 Streichquartett

D.3.2 Streichquartettstimmen

D.4 Vokalensemble

This section has not been translated yet; please refer to the manual in English.

D.4.1 SATB Partitur

D.4.2 SATB Partitur und automatischer Klavierauszug

D.4.3 SATB mit zugehörigen Kontexten

D.5 Vorlagen für alte Notation

This section has not been translated yet; please refer to the manual in English.

D.5.1 Transkription mensuraler Musik

D.5.2 Vorlage zur Transkription von Gregorianik

D.6 Jazz combo

This section has not been translated yet; please refer to the manual in English.

D.7 Lilypond-book-Vorlagen

This section has not been translated yet; please refer to the manual in English.

D.7.1 LaTeX

D.7.2 Texinfo

Anhang E Befehlsübersicht

Syntax

1 2 8 16

Erklärung

Tondauern

Beispiel



c4. c4..

Punktierung



c d e f g a b

Tonleiter



fis bes

Vorzeichen



\clef treble \clef bass

Notenschlüssel



\time 3/4 \time 4/4

Taktangaben



r4 r8

Pause



d ~ d

Bindebogen



`\key es \major`

Tonart

`note'`

Oktavierung

`note,`

Oktavierung nach unten

`c(d e)`

Legatobogen

`c\ (c(d) e\)`

Phrasierungsbogen

`a8[b]`

Balken

`<< \new Staff ... >>`

mehr Notensysteme

`c-> c-.`

Artikulationszeichen



`c2\mf c\s fz`

Dynamik

`a\< a a\!`

Crescendo

`a\> a a\!`

Decrescendo

`< >`

Noten im Akkord

`\partial 8`

Auftakt

`\times 2/3 {f g a}`

Triolen

`\grace`

Verzierungen

`\lyricmode { twinkle }`

Texteingabe

twinkle

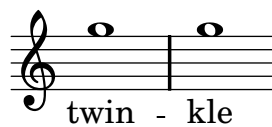
`\new Lyrics`

Textausgabe

twinkle

```
twin -- kle
```

Text-Trennstrich



```
\chordmode { c:dim f:maj7 }
```

Akkorde



```
\context ChordNames
```

Akkordsymbole drucken

 $C^{\circ} F^{\triangle}$

```
<<\{e f\} \\\{c d\}>>
```

Mehrstimmigkeit



```
s4 s8 s16
```

unsichtbare Pausen

Anhang F GNU Free Documentation License

Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ‘copyleft’, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The ‘Document’, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ‘you’.

A ‘Modified Version’ of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A ‘Secondary Section’ is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ‘Invariant Sections’ are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The ‘Cover Texts’ are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A ‘Transparent’ copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file

format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not ,Transparent‘ is called ,Opaque‘.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The ,Title Page‘ means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ,Title Page‘ means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled 'History', and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled 'History' in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the 'History' section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. In any section entitled 'Acknowledgments' or 'Dedications', preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgments and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section entitled 'Endorsements'. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section as 'Endorsements' or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to

the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled 'Endorsements', provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled 'History' in the various original documents, forming one section entitled 'History'; likewise combine any sections entitled 'Acknowledgments', and any sections entitled 'Dedications'. You must delete all sections entitled 'Endorsements'.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an 'aggregate', and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ‘or any later version’ applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

F.0.1 Anhang: Wie kann die Lizenz für eigene Dokumente verwendet werden

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.1
or any later version published by the Free Software Foundation;
with the Invariant Sections being list their titles, with the
Front-Cover Texts being list, and with the Back-Cover Texts being list.
A copy of the license is included in the section entitled 'GNU
Free Documentation License'
.
```

If you have no Invariant Sections, write *,with no Invariant Sections'* instead of saying which ones are invariant. If you have no Front-Cover Texts, write *,no Front-Cover Texts'* instead of *,Front-Cover Texts being list'*; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Anhang G Index der LilyPond-Befehle

(Index ist nicht vorhanden)

Anhang H LilyPond-Index

!		\grace.....	98
!	64	\ionian.....	83
,		\key.....	83
'	62	\locrian.....	83
,		\longa.....	70
,	62	\lydian.....	83
.		\major.....	83
.	70	\maxima.....	70
?		\mf.....	104
?	64	\minor.....	83
[\mixolydian.....	83
[.....	97	\mp.....	104
]		\oneVoice.....	81
]	97	\p.....	104
\		\partial.....	85
\!	105	\phrasingSlurDown.....	95
\<.....	105	\phrasingSlurNeutral.....	95
\>.....	105	\phrasingSlurUp.....	95
\\.....	76	\phrygian.....	83
\aeolian.....	83	\pp.....	104
\afterGrace.....	99	\ppp.....	104
\alternative notieren, etwa.....	110	\pppp.....	104
\arpeggio.....	109	\relative.....	65
\arpeggioBracket.....	110	\repeat.....	110
\arpeggioDown.....	110	\repeatTie.....	93, 111
\arpeggioNeutral.....	110	\rest.....	68
\arpeggioUp.....	110	\rfz.....	104
\bar.....	86	\sf.....	104
\breve.....	70	\sff.....	104
\cadenzaOff.....	88	\sfz.....	104
\cadenzaOn.....	88	\shiftOff.....	81
\clef.....	81	\shiftOn.....	81
\dorian.....	83	\shiftOnn.....	81
\dotsDown.....	71	\shiftOnnn.....	81
\dotsNeutral (Wiederherstellung des Standards.)	71	\skip.....	69
\dotsUp.....	71	\slurDashed.....	95
\dynamicDown.....	107	\slurDotted.....	95
\dynamicNeutral.....	107	\slurDown.....	95
\dynamicUp.....	107	\slurNeutral.....	95
\f.....	104	\slurSolid.....	95
\ff.....	104	\slurUp.....	95
\fff.....	104	\sp.....	104
\ffff.....	104	\spp.....	104
\fp.....	104	\startTrillSpan.....	108
\glissando.....	108	\stemDown.....	75
		\stemNeutral.....	75
		\stemUp.....	75
		\stopTrillSpan.....	108
		\super.....	129
		\tieDashed.....	94
		\tieDotted.....	94
		\tieDown.....	94
		\tieNeutral.....	94
		\tieSolid.....	94
		\tieUp.....	94
		\time.....	84
		\times.....	71
		\transpose.....	67
		\tupletDown.....	71
		\tupletNeutral.....	71
		\tupletUp.....	71

<code>\unfoldRepeats</code>	112
<code>\voiceFour</code>	81
<code>\voiceOne</code>	81
<code>\voiceThree</code>	81
<code>\voiceTwo</code>	81

|

.....	74
-------	----

~

~	92
---------	----

A

Abstand, zusätzlicher	54, 57
Abstände	58
Abstände füllen	53
Abstrich	101
<code>acciaccatura</code>	141
<code>addInstrumentDefinition</code>	142
<code>addQuote</code>	141
<code>afterGrace</code>	141
Akkolade	26
Akkordbezeichnungen	30
Akkorde	28, 30, 75
Akzent	101
Akzente	20
Akzidentien	18
Akzidenzen	63
Al niente	105
<code>allowPageTurn</code>	141
Altschlüssel	82
Anpassen der Ausgabe	10
Anschauen von Noten	12
Anzahl der Notenlinien einstellen	91
<code>applyContext</code>	141
<code>applyMusic</code>	143
<code>applyOutput</code>	141
<code>appoggiatura</code>	140
Arpeggio	109
Arpeggio	110
<code>arrow-head</code>	122
<code>arrow-head-markup</code>	122
Artikulation	20
Artikulationszeichen	21
Artikulationszeichen	101
<code>assertBeamQuant</code>	141
<code>assertBeamSlope</code>	141
Aufstrich	101
Auftakt	22
Auftakt	85
Ausklingen lassen	96
<code>auto-knee-gap</code>	97
<code>autochange</code>	142

B

B	18
Balance	2
Balken	22
Balken und Zeilenumbrüche	97
Balken zwischen Systemen	97

Balken, manuell	22, 97
<code>balloonGrobText</code>	142
<code>balloonText</code>	142
<code>bar</code>	142
<code>barCheckSynchronize</code>	74
Baritonschlüssel	82
<code>BarLine</code>	87
<code>barNumberCheck</code>	142
Bassschlüssel	82
<code>beam</code>	122
<code>Beam</code>	96, 114
<code>beam-markup</code>	122
Beginn eines Notensystems	88
<code>bendAfter</code>	142
Benutzung des Handbuchs	10
Beschriftung	101
Betrachten von Noten	12
Bezeichner	39, 46
<code>bigger</code>	122
<code>bigger-markup</code>	122
Binde- versus Legatobogen	20
<code>Bindebogen</code>	19, 20
Bindebogen	92
Blockkommentare	16
Bögen	20
Bögen, laissez vibrer	96
<code>bold</code>	122
<code>bold-markup</code>	122
<code>box</code>	122
<code>box-markup</code>	122
<code>bracket</code>	122
<code>bracket-markup</code>	122
Bratschenschlüssel	82
<code>breakable</code>	97
<code>breathe</code>	140
<code>BreathingSign</code>	107

C

C-Schlüssel	82
<code>caps</code>	122
<code>caps-markup</code>	122
<code>center-align</code>	123
<code>center-align-markup</code>	123
<code>char</code>	123
<code>char-markup</code>	123
<code>ChoirStaff</code>	89
<code>circle</code>	123
<code>circle-markup</code>	123
<code>clef</code>	140
<code>Clef</code>	83
Coda	101
<code>column</code>	123
<code>column-lines</code>	130
<code>column-lines-markup-list</code>	130
<code>column-markup</code>	123
<code>combine</code>	123
<code>combine-markup</code>	123
<code>Completion_heads_engraver</code>	74, 75
<code>compressMusic</code>	141
<code>concat</code>	123
<code>concat-markup</code>	123
Contexts	6
Converting from other formats	9

Crescendo..... 21, 106
 cueDuring..... 140

D

Dauer..... 70
 Daumenbezeichnung..... 101
 Decrescendo..... 21, 106
 defaultBarType..... 87
 Dialekt..... 10
 Dichte..... 2
 Dicke der Notenlinien einstellen..... 91
 Diminuendo..... 106
 dir-column..... 123
 dir-column-markup..... 123
 displayLilyMusic..... 140
 displayMusic..... 141
 Doppel-B..... 18
 Doppelkreuz..... 18
 Doppellinie..... 86
 Doppelpraller..... 101
 DotColumn..... 71
 Dots..... 71
 doubleflat..... 123
 doubleflat-markup..... 123
 DoublePercentRepeat..... 116
 DoublePercentRepeatCounter..... 116
 doublesharp..... 123
 doublesharp-markup..... 123
 draw-circle..... 123
 draw-circle-markup..... 123
 draw-line..... 123
 draw-line-markup..... 123
 dynamic..... 123
 dynamic-markup..... 123
 DynamicLineSpanner..... 56, 106, 107
 DynamicText..... 56, 107
 Dynamik..... 21, 104

E

Editor support..... 12
 Eigenschaften..... 10
 Eingabe von Noten parallel..... 91
 eingestrichenes C..... 13
 endSpanners..... 143
 Englische Begriffe..... 10
 Entfernen von Objekten..... 57
 epsfile..... 123
 epsfile-markup..... 123
 Erinnerungsvorzeichen..... 64
 Erweiterung von LilyPond..... 10
 Erweiterung von Wiederholungen..... 112
 Espressivo..... 101
 extra-offset..... 54, 57

F

FDL, GNU Free Documentation License..... 154
 featherDurations..... 141
 Fermate..... 101
 fill-line..... 123
 fill-line-markup..... 123
 filled-box..... 123

filled-box-markup..... 123
 finger..... 123
 finger-markup..... 123
 Fingering..... 104
 Fingersatz..... 21, 101, 103
 Fingerwechsel..... 103
 Flageolet..... 101
 flat..... 123
 flat-markup..... 123
 font-interface..... 127
 fontCaps..... 124
 fontCaps-markup..... 124
 fontsize..... 124
 fontsize-markup..... 124
 Forbid_line_break_engraver..... 75
 Formatierung von Brüchen..... 71
 Formatierung von Triolen..... 71
 fraction..... 124
 fraction-markup..... 124
 Franz. Violinschlüssel..... 82
 Fremdsprache..... 10
 fret-diagram..... 124
 fret-diagram-markup..... 124
 fret-diagram-terse..... 124
 fret-diagram-terse-markup..... 124
 fret-diagram-verbose..... 125
 fret-diagram-verbose-markup..... 125
 fromproperty..... 125
 fromproperty-markup..... 125
 Füllung..... 56
 Fußbezeichnung..... 101

G

ganze Note..... 14
 Gebrochene Akkorde..... 109
 Gedämpft..... 101
 general-align..... 125
 general-align-markup..... 125
 Glissando..... 108
 grace..... 142
 GraceMusic..... 100
 GrandStaff..... 88
 Groß- und Kleinschreibung..... 11, 16
 Großbuchstaben..... 11, 16

H

Hairpin..... 56, 105, 107
 halbe Note..... 14
 halign..... 125
 halign-markup..... 125
 hbracket..... 125
 hbracket-markup..... 125
 hcenter..... 125
 hcenter-in..... 125
 hcenter-in-markup..... 125
 hcenter-markup..... 125
 hspace..... 125
 hspace-markup..... 125
 huge..... 125
 huge-markup..... 125

I

includePageLayoutFile	141
Index	10
Instrumentengruppe	88
instrumentSwitch	142
interne Dokumentation	10
Intervalle	13
italic	126
italic-markup	126

J

Jargon	10
Justierung von Notensystemen	90
justified-lines	130
justified-lines-markup-list	130
justify	126
justify-field	126
justify-field-markup	126
justify-markup	126
justify-string	126
justify-string-markup	126

K

Kadenz	88
keepWithTag	143
KeyCancellation	84
KeySignature	84
killCues	143
Kirchentonarten	83
Klammer, geschweift	88
Klammer, vertikal	88
Klammer, Wiederholung	110
Klammern um Vorzeichen	64
Klaviersysteme	26
Kleinbuchstaben	11, 16
Kommentare	16
Kreuz	18

L

label	141
Laissez vibrer	96
LaissezVibrerTie	96
LaissezVibrerTieColumn	96
large	126
large-markup	126
larger	126
larger-markup	126
Lautstärke	21, 104
Layers	76
lead sheet	30
LedgerLineSpanner	64
left-align	126
left-align-markup	126
Legatobogen	20
Legatobögen	94
Legatobögen, Phrasierung	20
Lieder	28
Liedtext	28
LilyPond-book	9
lilypond-internals	10
line	126

line-markup	126
lookup	126
lookup-markup	126
lower	126
lower-markup	126
lowering text	129

M

magnify	126
magnify-markup	126
makeClusters	142
Marcato	101
markalphabet	126
markalphabet-markup	126
markletter	126
markletter-markup	126
Measure_grouping_engraver	85
MeasureGrouping	85
medium	126
medium-markup	126
mehrere Stimmen	27
Mehrstimmigkeit	27, 76
Melisma	29
Melismen	29
Metrum	84
Mezzosopranschlüssel	82
Mikrotöne	64
Modus	83
Mordent	101
moving text	129
musicglyph	126
musicglyph-markup	126
musicMap	143
Musikalischer Ausdruck	23
Musiksymbole	2

N

N-tolen	22, 71
N-tolen, Formatierung	71
natural	126
natural-markup	126
Niente, al	105
noPageBreak	142
noPageTurn	142
normal-size-sub	126
normal-size-sub-markup	126
normal-size-super	127
normal-size-super-markup	127
normal-text	127
normal-text-markup	127
normale Abstände	3
normale Rhythmen	3
normalsize	127
normalsize-markup	127
Notation von Systemelementen	81
note	127
note-by-number	127
note-by-number-markup	127
note-markup	127
Note_heads_engraver	74
NoteCollision	79, 81
NoteColumn	81

NoteHead	64
Notenausgabe	12
Notenbezeichnungen, andere Sprachen	65
Notenbezeichnungen, Deutsch	63
Notenbezeichnungen, Standard	63
Notenköpfe, Stile	76
Notenlänge	70
Notenlinien, Anzahl	91
Notenlinien, Dicke	91
Notensatz	5
Notenschlüssel	15
Notenschlüssel	82
Notensysteme, mehrere	88
Notensysteme, Modifikation	90
null	127
null-markup	127
number	127
number-markup	127

O

octave	141
Offen	101
Oktavenüberprüfung	67
oldaddyrics	141
on-the-fly	127
on-the-fly-markup	127
Optimierung von Abständen	58
Optischer Ausgleich	2
Orgelpedalbezeichnung	101
Ossia	90
override	127
override-lines	130
override-lines-markup-list	130
override-markup	127
overrideProperty	140

P

pad-around	127
pad-around-markup	127
pad-markup	127
pad-markup-markup	127
pad-to-box	127
pad-to-box-markup	127
pad-x	127
pad-x-markup	127
padding	56
page-ref	127
page-ref-markup	127
pageBreak	142
pageTurn	141
Parallele Notation, Eingabe	91
parallelMusic	141
parenthesize	142
partcombine	142
Partitur	88
Pause	15
Pausen	68
PDF-Datei	12
PercentRepeat	116
PercentRepeatCounter	116
PercentRepeatedMusic	116
Phrasierungsbögen	20, 95

PhrasingSlur	95
PianoStaff	109
pipeSymbol	74
pitchedTrill	141
Platzhalternoten	69
Polyphonie	76
Portato	101
postscript	127
postscript-markup	127
Praller	101
Prallermordent	101
Prima volta	110
properties	10
Prozent-Wiederholungen	115
punktierte Noten	14
Punktierung	70
put-adjacent	128
put-adjacent-markup	128

Q

Quarte	13
quoteDuring	142

R

r	68
raise	128
raise-markup	128
raising text	129
Relativ	65
Relative Oktavbestimmung	65
removeWithTag	140
repeatCommands	87, 113
RepeatedMusic	114
RepeatSlash	116
resetRelativeOctave	141
Rest	69
RestCollision	81
Rhythmische Aufteilungen	71
right-align	128
right-align-markup	128
rightHandFinger	143
roman	128
roman-markup	128
rotate	128
rotate-markup	128
rotated text	127
Running LilyPond	9

S

s	69
sans	128
sans-markup	128
Scheme	10
Schlüssel	82
Schnipsel	10
Schriftart	2
score	128
Score	85
score-markup	128
scoreTweak	142
Script	103

Seconda volta	110
Segno	101
semiflat	128
semiflat-markup	128
semisharp	128
semisharp-markup	128
sesquiflat	128
sesquiflat-markup	128
sesquisharp	128
sesquisharp-markup	128
sharp	129
sharp-markup	129
shiftDurations	142
simple	129
simple-markup	129
Skip	69
SkipMusic	70
slashed-digit	129
slashed-digit-markup	129
Slur	95
small	129
small-markup	129
smallCaps	129
smallCaps-markup	129
smaller	129
smaller-markup	129
Song-Blatt	30
Sopranschlüssel	82
spacingTweaks	142
SpanBar	87
Sprache	10
Staccatissimo	101
Staccato	20, 101
Staff	69, 87
StaffGroup	88
StaffSymbol	91
Stem	75
stemLeftBeamCount	97
stemRightBeamCount	97
StemTremolo	114, 115
stencil	129
stencil-markup	129
Stimmen, mehrere in einem System	27
Stimmgruppe	88
Stimmwechsel zwischen Systemen, manuell	26
strut	129
strut-markup	129
sub	129
sub-markup	129
Subbassschlüssel	82
subdivideBeams	97
Suche im Handbuch	10
super	129
super-markup	129
Symbole, systemweit	81
System, Chor	88
Systeme, mehrere	88
Systemelemente	81
Systemgruppe	88
SystemStartBar	89
SystemStartBrace	89
SystemStartBracket	89
systemStartDelimiter	89
Systemwechsel, manuell	26

T

tag	142
Taktangabe	15
Taktangabe	84
Takte verkürzen	85
Taktlinien	86
Taktüberprüfung	74
Taktweise Wiederholungen	115
teeny	129
teeny-markup	129
Tenorschlüssel	82
Tenorschlüssel, Chor	82
Tenuto	101
Terminologie	10
text	129
Text	28
text-interface	127
text-markup	129
TextScript	103
The Feta font	126
Tie	94
tied-lyric	129
tied-lyric-markup	129
TimeScaledMusic	73
TimeSignature	85
Timing_translator	85
tiny	129
tiny-markup	129
tocItem	140
Tonart	83
Tonart, Einstellung von	18
Tondauer	14
Tonhöhenbezeichnungen	62
Tonleiter	13
translate	129
translate-markup	129
translate-scaled	129
translate-scaled-markup	129
translating text	129
transparent	130
transparent-markup	130
transparente Objekte	57
Transponieren	67
transposedCueDuring	140
TransposedMusic	68
transposition	141
Transposition	67
Tremolobalken	114
tremoloFlags	115
Tremolozeichen	115
triangle	130
triangle-markup	130
Triller	101, 107
Triller mit Tonhöhe	108
TrillSpanner	108
Triolen	22, 71
TupletBracket	73
TupletNumber	73
tupletNumberFormatFunction	71
tweak	143
typewriter	130
typewriter-markup	130
Typographie	3, 5

U

Überbindung.....	92
Überspringen von Zeichen.....	69
UnfoldedRepeatedMusic.....	114
unfoldRepeats.....	142
unsichtbare Objekte.....	57
Unsichtbare Pausen.....	69
Unterstrich zur Silbenverlängerung.....	29
Updating files with convert-ly.....	31, 51
upright.....	130
upright-markup.....	130

V

Varcodea.....	101
Variable.....	46
Variablen.....	10, 39
vcenter.....	130
vcenter-markup.....	130
Veränderungen von Abständen.....	58
verbatim-file.....	130
verbatim-file-markup.....	130
Verschachtelte Musik.....	91
Versetzungszeichen.....	18, 63
Versetzungszeichen, Erinnerung.....	64
Versetzungszeichen, Vierteltöne.....	64
Versetzungszeichen, Warnung.....	64
Versionsnummern.....	31
Verstecken von Objekten.....	57
Verzierungen.....	23, 98
Viertelnote.....	14
Vierteltöne.....	64
Violinschlüssel.....	82
Voice.....	69, 76, 77, 105
Volta.....	110
Volta und Überbindung.....	93
VoltaBracket.....	114
VoltaRepeatedMusic.....	114
Vorhalt.....	23, 98
Vorschlag.....	23, 98

Vorzeichen.....	18, 83
Vorzeichen in Klammern.....	64
Vorzeichen, Deutsch.....	63
Vorzeichen, Erinnerung.....	64
Vorzeichen, Vierteltöne.....	64

W

Warnungsvorzeichen.....	64
Wechsel zwischen Systemen, manuell.....	26
whichBar.....	87
whiteout.....	130
whiteout-markup.....	130
Wiederholung und Überbindung.....	93
Wiederholung, mehrdeutig.....	112
Wiederholungen.....	110
Wiederholungszeichen.....	86
with-color.....	130
with-color-markup.....	130
with-dimensions.....	130
with-dimensions-markup.....	130
with-url.....	130
with-url-markup.....	130
withMusicProperty.....	141
wordwrap.....	130
wordwrap-field.....	130
wordwrap-field-markup.....	130
wordwrap-lines.....	130
wordwrap-lines-markup-list.....	130
wordwrap-markup.....	130
wordwrap-string.....	130
wordwrap-string-markup.....	130

Z

Zahl der Notenlinien einstellen.....	91
Zeichen.....	101
Zeilenkommentare.....	16
Ziernoten.....	98
zusätzlicher Abstand.....	57