UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Eingabe und Darstellung von Noten für die musikalische Ausbildung von Kindern

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von

Svenja Nußbaum

Erstgutachter:	Prof. Dr. Stefan Müller Institut Computervisualistik, Arbeitsgruppe Computergrafik
Zweitgutachter:	Kevin Keul, M.Sc. Institut Computervisualistik, Arbeitsgruppe Computergrafik

Koblenz, im März 2016

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Zusammenfassung

Bei der musikalischen Grundausbildung von Kindern und Jugendlichen besteht eine spezielle Herausforderung darin, den Kindern das Notenlesen und -schreiben näher zu bringen. Bei der Ausbildung von jungen Nachwuchssängerinnen und -sängern eines Chores ist es zudem wichtig sie damit vertraut zu machen niedergeschriebene Noten direkt in Töne umzusetzen.

Eine interessante Idee ist es, den Kindern während des Unterrichts die Töne auf einem Klavier oder Keyboard vorzuspielen und diese gleichzeitig auf einem Bildschirm in Notenschrift visuell darzustellen.

Ziel dieser Bachelorarbeit ist die Implementierung eines solchen Systems, welches das Einspielen von Noten mit Hilfe eines MIDI-fähigen Keyboards und die anschließende visuelle Wiedergabe der Noten in Notenschrift umsetzt. Die prototypische Anwendung arbeitet in drei Schritten. Sie erhält über ein angeschlossenes Keyboard die Noten in Form von MIDI-Datensätzen als Eingabe. Diese MIDI-Informationen werden dann in das MusicXML-Format überführt. Ausgehend von dieser Notation in MusicXML wird abschließend die visuelle Ausgabe in Form von Notenschrift generiert und angezeigt.

Abstract

A special challenge of the basic musical education of children is to give them an understanding of reading and writing musical scores. During the training of young choristers it is furthermore important to educate them in directly transforming the written scores into sounds.

Therefore it is an interesting idea to play the sounds to the children via piano or keyboard and simultaneously present them on a screen in musical notation.

The aim of this bachelor thesis is the implementation of such a system that allows to enter scores using a MIDI-compatible keyboard and then depicting these as musical notation. The prototype of the application operates in three steps. It receives the musical scores via keyboard in form of MIDI-datasets. These MIDI-information are converted to the MusicXML-format. Based on this MusicXML-notation the software finally generates and displays the visual output.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die musikalische Bildung und Ausbildung von Kindern und Jugendlichen birgt viele Herausforderungen. Besonders die Umsetzung von theoretischen Grundlagen in der Praxis, beispielsweise in einem Chor, stellt sich für viele jüngere Musikerinnen und Musiker als schwierig heraus. Auch im St. Martins-Chor Bad Ems[1] wird viel Zeit in die Forderung und Förderung der Kinder des Chores investiert. Von dort stammt der Anstoß zu diesem Projekt. Die Planung und Umsetzung fand daher auch in enger Zusammenarbeit mit dem Leiter des Chores, Herrn Lutz Brenner, statt.

Dieser unterrichtet neben seiner Funktion als Chorleiter die jungen Nachwuchsänger und -sängerinnen in den musikalischen Grundlagen, die für eine erfolgreiche Laufbahn in einem Chor notwendig sind. Dazu zählt, die Kinder zunächst mit dem Notenlesen und -schreiben vertraut zu machen und ihnen die praktische Umsetzung der geschriebenen Noten in Form von Gesang näher zu bringen. Sie sollen zum Beispiel lernen die Noten direkt „vom Blatt“ zu singen. Das bedeutet, dass die vom Notenblatt abgelesenen Notenfolgen ohne Zuhilfenahme eines Instruments direkt in Tönen umgesetzt werden. Vor allem für junge Sängerinnen und Sänger bietet dies eine spezielle Herausforderung. Für diesen Unterricht suchte Herr Brenner nach einer Softwarelösung, die das veraltete und in seinen Funktionen eingeschränkte System der *„Lichtorgel“* ersetzen und um einige Funktionen erweitern sollte. Die Erstellung einer solchen Anwendung nach den Vorstellungen des Chorleiters ist der Hintergrund dieser Arbeit.

1.2 Aktuelle Situation beim St. Martins-Chor

Aktuell wird die musikalische Grundausbildung der Nachwuchssänger und Nachwuchssängerinnen beim St. Martins-Chor durch den Einsatz einer sogenannten *„Lichtorgel“* (siehe Abb. 1) unterstützt.

Die Lichtorgel besteht aus einer Klaviatur mit insgesamt 29 Tasten (17 weiße und 12 schwarz) und einem Anzeigegerät. Die beiden Teile sind durch ein Kabel miteinander verbunden. Auf dem Anzeigegerät sind fünf Notelinien sowie ein Violinschlüssel eingezeichnet. Außerdem sind die Töne g bis a″ als ganze Noten im Violinschlüssel eingetragen. Wird eine Taste auf der Klaviatur gedrückt, so leuchtet die gespielte Note auf der Anzeige auf (siehe Abb. 2). Wird die Taste wieder losgelassen, erlischt auch das zugehörige Licht auf der Anzeige. Dabei können theoretisch beliebig viele Tasten gleichzeitig betätigt werden. Wegen der schon etwas veralteten Technik, werden die einzelnen Leuchten jedoch mit jeder weiteren, die hinzukommt schwächer.

[1] Homepage: `http://www.st-martins-chor.de/`

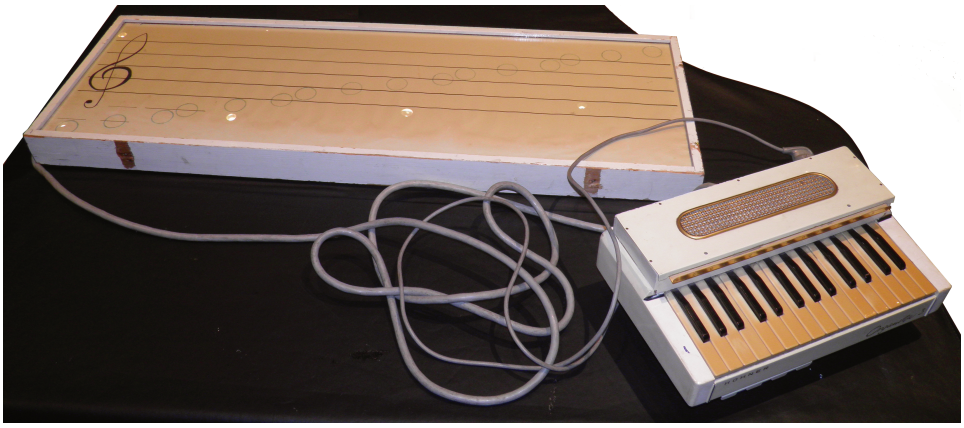


Abbildung 1: Lichtorgel bestehend aus einem Anzeigeelement und einer Klaviatur

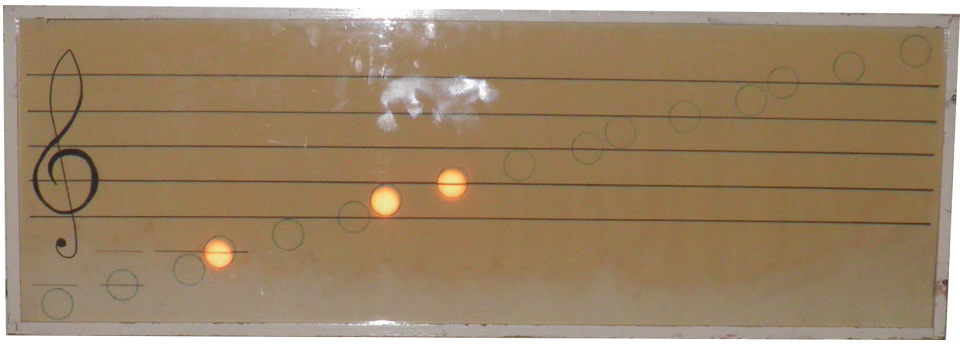


Abbildung 2: Anzeige der Lichtorgel bei Betätigung mehrerer Tasten

Die Darstellungsmöglichkeiten der Lichtorgel sind aufgrund ihrer Funktionsweise sehr beschränkt. Sie kann zwar 16 verschiedene Tonhöhen (die Stammtöne[2]) und entsprechende Tonabstände anzeigen. Tonlängen oder Vorzeichen werden hingegen ignoriert. Die Wiedergabe von Akkorden, Tonarten, Pausen oder Takten bzw. Taktarten ist ebenfalls nicht möglich.

Diese Einschränkungen bei der Ausbildung der Kinder sollen durch die hier entwickelte Softwarelösung beseitigt werden.

1.3 Ziele

Schwerpunkt dieser Arbeit ist die konzeptionelle Planung und darauffolgende prototypische Implementation eines Systems, welches das Einspielen von Noten über ein MIDI-Keyboard und die anschließende optische Wiedergabe der Noten in Form von Notenschrift auf dem Bildschirm echt-

[2] Stammton = Ton ohne Vorzeichen

zeitfähig ermöglicht. Die Notenschrift soll dabei neben den Noten selbst die Darstellung von Notenschlüsseln, Taktarten und Takten beinhalten.

Zu Beginn sollen die an die Anwendung gestellten Anforderung erhoben und klar definiert werden. Auf Basis der Anforderungsanalyse soll dann ein Konzept für das System entworfen werden. Dieses Konzept soll im Anschluss während einer Implementierungsphase in einen funktionsfähigen Prototypen umgesetzt werden. Im Mittelpunkt soll hierbei die Realisierung der korrekten grafischen Wiedergabe der eingespielten Töne stehen.

Die Anwendung soll für die musikalische Bildung und Ausbildung von Kindern verwendet werden. Das Programm soll also entsprechend im Hinblick auf den späteren Einsatz im Unterricht gestaltet werden. Beispielsweise soll die Benutzeroberfläche übersichtlich und schlicht gehalten werden, damit die Aufmerksamkeit der Schülerinnen und Schüler nicht von dem wesentlichen Aspekt, der Darstellung der Noten, abgelenkt wird.

1.4 Anforderungen

Zu Beginn des Projektes wurde in Zusammenarbeit mit dem Leiter des St. Martins-Chores, Lutz Brenner, eine Anforderungsanalyse durchgeführt. Die Ergebnisse dieser Analyse sollen im Folgenden aufgezeigt werden. Die Anforderungen gliedern sich in vier Gruppen.

Die erste Gruppe bezieht sich auf das System insgesamt. Die Bedienung des Systems muss intuitiv sein und darf keine lange Einarbeitungszeit erfordern. Dazu ist es wichtig die Funktionen des Systems auf ein erforderliches Minimum zu beschränken. Die Eingabe der darzustellenden Noten in das System muss über ein MIDI-fähiges Keyboard und entsprechend im MIDI-Datenformat getätigt werden. Dabei soll das System zwei unterschiedliche Eingabe- oder auch Darstellungsmodi unterstützen:

1. den Tonfolgemodus, bei dem die eingespielten Töne als Tonfolge dargestellt werden.

2. den Akkordmodus, bei dem die eingespielten Töne als Akkorde dargestellt werden. Dieser Modus ermöglicht einen schrittweisen Akkordaufbau.

Optional sind eine Möglichkeit zur Speicherung der eingegebenen Daten, sodass diese anschließend wieder geladen werden können und die Speicherung der eingegebenen Daten in einem Bildformat oder als PDF-Datei. Denkbar wäre auch eine Komponente zur akustischen Wiedergabe der eingegebenen Noten in das System einzubinden.

Die zweite Gruppe befasst sich mit der optischen Wiedergabe des benötigten Notensystems. Im Ausgabefenster der Anwendung sollen unabhängig davon, ob bereits Noten eingegeben wurden oder nicht, mindestens ein bis zwei Notenzeilen angezeigt werden. Dazu muss neben den fünf Notenlinien eines jeden Notensystems zu Beginn jeder Notenzeile ein Notenschlüssel (Violin- oder Bassschlüssel) abgebildet werden. Jede Notenzeile soll zudem direkt im Anschluss an den Notenschlüssel eine Taktangabe beinhalten. Optional ist die Wiedergabe von Tonarten durch die Darstellung von Vorzeichen, welche zwischen Notenschlüssel und Taktangabe eingefügt würden, in einer jeden Notenzeile.

Die dritte Anforderungsgruppe geht auf die optische Wiedergabe der eingegebenen Noten ein. Die Ausgabe der Notendarstellung erfolgt wahlweise auf einem Bildschirm oder mit Hilfe eines Beamers. Dabei muss die Tonhöhe der Noten korrekt dargestellt werden. Außerdem soll die entsprechende Tondauer ermittelt und ebenfalls korrekt angezeigt werden. Einzelne Noten, beispielsweise die zuletzt eingegebene Note, sollen wieder aus der Anzeige gelöscht werden können. Neben der Tonhöhe soll auch die Wiedergabe von Versetzungszeichen vor den Noten unterstützt werden. Des Weiteren sollen Pausen darstellbar sein.

Die letzte Gruppe bezieht sich auf eine optionale Benutzeroberfläche. Es muss möglich sein Informationen, die nicht in den MIDI-Daten codiert werden (wie Notenschlüssel, Taktarten, Tempi etc.), in das System einzugeben. Eine Möglichkeit dies umzusetzen ist eine grafische Benutzeroberfläche. Die Benutzeroberfläche soll übersichtlich sein und darf die Aufmerksamkeit des Benutzers und der unterrichteten Kinder nicht von der eigentlichen Anwendung ablenken. Sie sollte schlicht und kompakt gehalten sein und sollte idealerweise nur die für die Benutzung der Anwendung unbedingt notwendigen Einstellungsoptionen beinhalten.

1.5 Existierende Softwarelösungen

Auf dem Markt gibt es bereits eine große Bandbreite an Software, die Lösungen für das Problem der Noteneingabe via MIDI-Keyboard und der anschließenden visuellen Wiedergabe der Noten in Notenschrift bieten. Nahezu alle von diesen Programmen sind sogenannte Notensatzprogramme. Diese wurden mit der Absicht entwickelt eine inhaltlich korrekte und gleichzeitig optisch ansprechend Darstellung von (umfangreicheren) Musikstücken in Form von Notenblättern und ganzen Partituren[3] zu er-

[3] „zusammenfassende Niederschrift eines vielstimmigen Musikstückes, jeweils in Einzelstimmen Takt für Takt untereinander“ (Quelle: `http://www.wissen.de/fremdwort/partitur`)

möglichen. Die Notensatzprogramme *MuseScore*[4], *Finale*[5] und *Capella*[6] unterstützen beispielsweise die Eingabemethode via MIDI-Keyboard bei der Erzeugung der Notenschrift.

Sie eignen sich jedoch kaum für den Einsatz in der musikalischen Ausbildung von Kindern. Ein Problem ist, dass sie, wie schon erwähnt, für einen anderen Zweck entwickelt worden sind. Sie bieten zahlreiche, komplexe Einstellungsmöglichkeiten, die für das Erstellen von vollständigen Notensätzen unentbehrlich, für den Unterrichtseinsatz jedoch ungeeignet sind. Die Benutzeroberfläche der Programme (Abb. 3 zeigt ein Beispiel) ist sehr umfangreich, meist wenig intuitiv und kann einen Nutzer leicht überfordern. Durch eine Reduzierung der Einstellungsmöglichkeiten auf ein Minimum soll die hier entwickelte Anwendung diese Probleme ausschließen und den Einsatz der Software im Unterrichtsalltag so angenehm und einfach wie möglich gestalten.

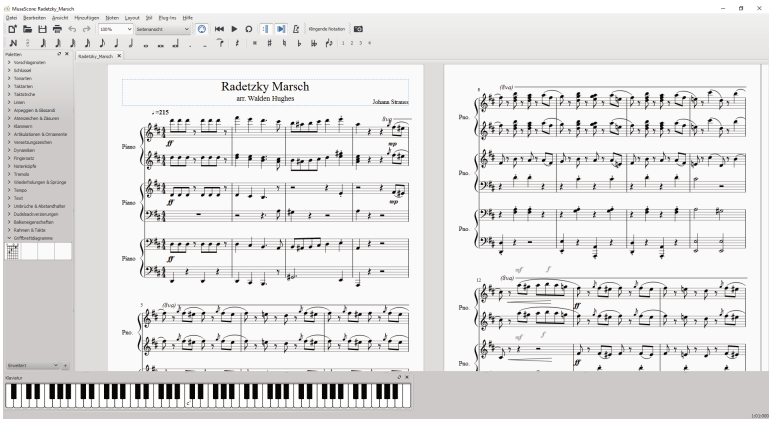


Abbildung 3: Screenshot des Programms „MuseScore“

Des Weiteren ist für die Eingabe der Noten in diesen Programmen teilweise ein ständiger Wechsel zwischen dem MIDI-Keyboard und der Maus sowie der Tastatur des PCs notwendig. Dies ist zum Beispiel bei der Verwendung der freien Software *MuseScore* notwendig, um die Dauer einer Note einzugeben. Das Programm unterstützt keine automatische Erkennung von Tondauern bei der Eingabe der Töne über ein MIDI-Keyboard. Auch für das Einspielen von Akkorden in *MuseScore* ist ein zusätzlicher Tastendruck auf der PC-Tastatur notwendig.

Der Wechsel zwischen den verschiedenen Eingabemedien (Maus, Tastatur, MIDI-Keyboard) soll durch die automatische Erkennung von möglichst vielen Informationen (z.B. Tondauer, Akkorde) weitestgehend eingedämmt werden.

[4]Homepage: `https://musescore.org/de`
[5]Homepage: `http://www.finalemusic.com/`
[6]Homepage: `http://www.capella.de/de/index.cfm/`

1.6 Das grobe Konzept des Systems

Vor Beginn der genaueren Recherchearbeit wurde ein grobes Konzept der zu entwickelnden Anwendung erstellt. Dieses basiert auf den Anforderungen, die in Abschnitt 1.4 beschrieben wurden, und stellt die angedachte Verarbeitungspipeline der Musikdaten durch das Programm dar.

Die Anwendung soll zunächst die Eingaben, die über das MIDI-Keyboard getätigt werden, einlesen. Die empfangenen MIDI-Nachrichten sollen dann in ein weiterverarbeitbares Daten-Format umgewandelt und zwischengespeichert werden. Hierfür wurde das MusicXML-Datenformat angedacht. Vorteil dieses Formats wäre, dass die Daten nicht nur von der Anwendung selbst, sondern auch von schon bestehender Software (siehe Abschnitt 1.5) gelesen und wiedergegeben werden können. Zum Schluss folgt die Wiedergabe dieser gespeicherten Daten auf dem Bildschirm. Parameter, die nicht aus den MIDI-Daten abgeleitet werden können, wie der Notenschlüssel, die Taktart oder auch das Tempo, müssen dabei vom Benutzer zusätzlich zum Beispiel über eine grafische Benutzerschnittstelle eingegeben werden.

1.7 Aufbau der Arbeit

Nach dieser umfangreichen Einleitung folgt in Kapitel 2 die Beschreibung der Recherchephase. Darunter fallen die Erläuterung der für den Kontext wichtigen musikalischen Grundlagen sowie die Vorstellung der im Laufe des Projektes verwendeten Datei-Formate *MIDI* und *MusicXML* und der eingesetzten Softwarebibliotheken. Kapitel 3 beinhaltet darauf aufbauend Ausführungen zum Implementierungsprozess mit einem kurzen Überblick und nachfolgenden Beschreibungen der einzelnen Teile des Systems. Nach Abschluss der Implementierung wurde eine Anwenderbefragung in Zusammenarbeit mit Herrn Lutz Brenner als Experten durchgeführt. Aufbau und Ergebnisse dieser Evaluation sollen Thema des 4. Kapitels sein. Letztlich wird in Kapitel 5 ein Fazit gezogen und ein Ausblick auf Möglichkeiten zur Weiterentwicklung des Systems gegeben.

2 Recherche

Nachdem das grobe Konzept des Prototypen (siehe Abschnitt 1.6) erstellt wurde und feststand welche Hilfsmittel und Datenformate erforderlich waren, begann der in diesem Kapitel thematisierte Rechercheprozess. Während dieser Phase wurden zunächst die nötigen Informationen zum Fachgebiet Musik gesammelt. Dazu zählt zum einen, dass Begriffe, wie beispielsweise „Note“ oder „Notenwert“, eindeutig definiert werden, zum anderen, dass Wissen zu weiteren musikalischen Grundlagen, die zum Verständnis des Kontextes wichtig sind, erworben wird. Anschließend wurde

das MIDI-Format näher betrachtet und ein passendes Format zum Zwischenspeichern und Übertragen der Musikdaten innerhalb der Anwendung gesucht. Für letzteres zeigte sich das MusicXML-Datenformats als geeignet. Ein Vorteil dieses Formats ist, dass es inzwischen von vielen (Notensatz-) Programmen unterstützt wird und so die eingespielten Daten auch von anderer Software gelesen werden kann. Letztlich mussten Softwarebibliotheken ausgewählt und eingebunden werden um Arbeiten zu übernehmen, die im Rahmen dieser Arbeit nicht selbst implementiert werden sollten. Zu diesen Aufgaben zählen das Einlesen der MIDI-Daten, die Bereitstellung von Funktionen zum Schreiben und Lesen von XML-Dateien, die Generierung der Texturen zur visuellen Darstellung der Musikdaten sowie die Bereitstellung von Funktionen zur Erstellung einer grafischen Benutzeroberfläche.

2.1 Musikalische Grundlagen und Begriffsdefinitionen

Im Weiteren werden die für dieses Projekt benötigten musikalischen Grundlagen kurz beschrieben und die wichtigsten Begriffe definiert. Einige dieser Begriffe haben je nach betrachtetem Teilgebiet der Musik leicht unterschiedliche Bedeutungen. Hier wird jeweils nur der Sinn im Rahmen des in dieser Arbeit geltenden Kontextes erwähnt. Außerdem bauen verschiedene Begriffsdefinitionen gegenseitig aufeinander auf. Von jeweiligen Querverweisen innerhalb dieses Abschnitts wurde daher abgesehen. Die gesamte Passage basiert auf den Ausführungen von Wieland Ziegenrücker [Zie88].

Note

Die *Note* ist das wichtigste Symbol, das bei der schriftlichen Aufzeichnung von Musik verwendet wird. Durch die Platzierung und äußere Form lässt sich auf Höhe und Dauer des dargestellten Tons schließen. Eine *Note* besteht aus einem Notenkopf, der entweder hohl oder ausgefüllt ist und (abgesehen von den ganzen Noten) einem Notenhals. Der Notenhals wird rechts vom Kopf nach oben notiert, wenn der Ton unterhalb der dritten Notenlinie (der Mittellinie) im Notensystem liegt, ansonsten wird der Hals links vom Kopf nach unten notiert. Am Notenhals können jeweils rechts am Ende ein oder mehrere sogenannte Fähnchen angebracht sein. Folgen mehrere *Noten* mit Fähnchen aufeinander, so werden diese gewöhnlich stattdessen mit Balken verbunden.

Notensystem

(oder auch Notenzeile)
Ein *Notensystem* besteht aus fünf waagerechten, parallelen Linien, die je-

weils den gleichen Abstand zueinander haben. Es fungiert als eine Art Koordinatensystem für Noten. In vertikaler Richtung werden die Tonhöhen aufgetragen, während die horizontale Richtung die zeitliche Abfolge der Noten wiedergibt. Die Noten werden sowohl zwischen, als auch auf den Linien platziert, um Tonhöhen darzustellen. Die Notenlinien werden immer von unten nach oben gezählt.

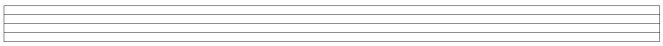


Abbildung 4: Ein Notensytem bestehend aus fünf Notenlinien

Sollte ein Ton aufgrund seiner Höhe überhalb oder unterhalb der fünf Linien des *Notensystems* geschrieben werden müssen, so werden sogenannte *Hilfslinen* als Orientierungshilfe notiert. Wird ein *Notensystem* mit Violinschlüssel und eines mit Bassschlüssel als sogenannte Akkolade direkt untereinander notiert, so wird dies als *Klaviersystem* bezeichnet.

Notenschrift

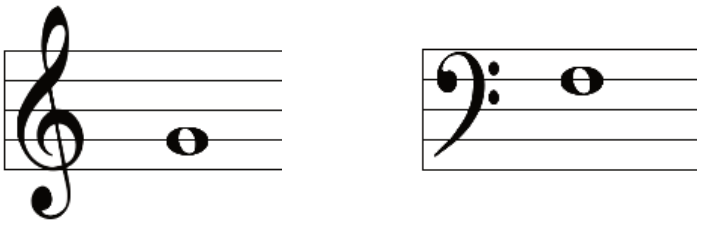
Bezeichnet die schriftliche Darstellung von Tönen und Musik durch eigens dafür entwickelte Schriftzeichen. Dabei wird ein Ton durch ein Notenzeichen dargestellt. Die Form dieses Zeichens codiert die Tondauer, während dessen (vertikale) Position in einem Notensystem die Tonhöhe angibt. Zur *Notenschrift* gehören außerdem die Notation von Notenschlüsseln, Takten und Taktarten, Tonarten, Versetzungszeichen etc.

Notenschlüssel

Das Ablesen der Tonhöhe kann erst geschehen, wenn eine der Notenlinien im Notensystem als Basislinie definiert wurde. Dies geschieht durch die sogenannten *Notenschlüssel*. Der *Violinschlüssel* (oder G-Schlüssel) definiert die Lage des g der eingestrichenen Oktave auf die zweite Notenlinie. Diese wird durch den „Wirbel“ des Schlüssels eingeschlossen. Ein weiterer Schlüssel ist der *Bassschlüssel* (oder F-Schlüssel). Er definiert die Lage des f der kleinen Oktave auf die vierte Notenlinie. Diese wird durch den Doppelpunkt eingeschlossen. Alle weiteren *Notenschlüssel* sind hier nicht weiter von Belang.

Tonhöhe

Die *Tonhöhe* wird durch die Notennamen angegeben. Die in westlichen Gebieten gebräuchlichen Notennamen lauten: c, d, e, f, g, a, h (, c). Diese bilden die sogenannte *Stammtonreihe*. Da es mehr als sieben verschiedene Töne gibt, wird diese Stammtonreihe mehrfach aneinandergereiht, um alle Töne darstellen zu können. Dadurch entstehen verschiedene Oktavbereiche.



(a) Violinschlüssel mit Note g′

(b) Bassschlüssel mit Note f

Abbildung 5: Notenschlüssel

Oktave

Eine *Oktave* umfasst den Abstand von der ersten zur achten Tonstufe einer Stammtonreihe. Die Klaviatur eines großen Flügels umfasst 52 Stammtöne. So werden neun verschiedene *Oktavbereiche* festgelegt:

Oktavbereich	Notation
fünfgestrichene Oktave (höchste Oktave)	nur c′′′′
viergestrichene Oktave	c′′′ bis h′′′
dreigestrichene Oktave	c′′′ bis h ′′′
zweigestrichene Oktave	c′′ bis h′′
eingestrichen Oktave	c′ bis h′
kleine Oktave	c bis h
große Oktave	C bis H
Kontra-Oktave	‚C bis ‚H
Subkontra-Oktave	„A und „H

Tabelle 1: Die verschiedenen Oktavbereiche und die jeweilige Notationskonvention für die jeweiligen Stammtöne (entnommen aus [Zie88])

Versetzungszeichen/Vorzeichen

Durch die sogenannten *Versetzungszeichen* (siehe Abb. 6), die direkt vor einem Stammton notiert werden, können diese Stammtöne jeweils um einen Halbton „erhöht“ oder „herabgesetzt“ werden. Zum einfachen Erhöhen wird das Kreuz, zum doppelten Erhöhen (Erhöhung um zwei Halbtonschritte) das Doppel-Kreuz verwendet. Äquivalent wird zum Herabsetzen das einfache b und das Doppel-b genutzt. *Versetzungszeichen* können durch das sogenannte Auflösungszeichen wieder aufgehoben werden. Das *Versetzungszeichen* wird hierbei genau im Raum (auf der Linie oder im Zwischenraum) der betroffenen Note geschrieben und gelten nur im jeweiligen

Takt und Oktavbereich. *Versetzungszeichen* haben keine Hilfslinien. *Vorzeichen* haben im Grunde dieselbe Funktion wie *Versetzungszeichen*. Im Gegensatz zu diesen werden *Vorzeichen* jedoch am Anfang einer Notenzeile vor der Taktangabe eingezeichnet und kennzeichnen die Tonart (hier nicht weiter ausgeführt). Sie gelten für das gesamte Musikstück und alle Oktavbereiche.

(a) Doppel-b (b) b (c) Auflösungszeichen (d) Kreuz (e) Doppel-Kreuz

Abbildung 6: Versetzungszeichen

Quintenzirkel

Im *Quintenzirkel* werden die Dur- und Moll-Tonarten mit jeweils gleichen Vorzeichen nach einem bestimmten Schema angeordnet. Aus dem *Quintenzirkel* lassen sich die Art und Anzahl der Vorzeichen für die jeweilige Tonart ablesen.

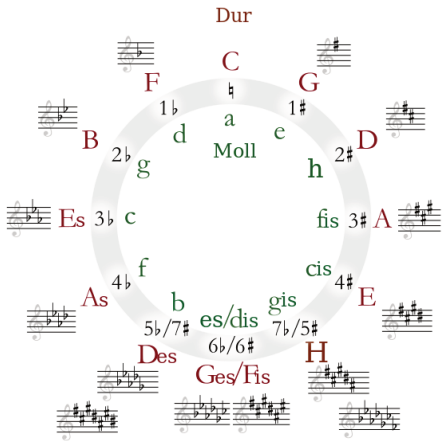


Abbildung 7: Quintenzirkel[7]

[7]Quelle: `https://de.wikipedia.org/wiki/Quintenzirkel`

Tondauer und Pausen

Die *Tondauer* lässt sich aus der Gestalt der einzelnen Noten sowie den Angaben zum Tempo des Stücks herleiten. Die *Notenwerte* (Ganze, Halbe, Viertel usw.) geben dabei jeweils nur relative Dauern an.

Name	Note	Beschreibung	zugehörige Pause
Ganze		hohler Notenkopf ohne Notenhals	
Halbe		hohler Notenkopf mit Notenhals	
Viertel		ausgefüllter Notenkopf mit Notenhals	
Achtel		ausgefüllter Notenkopf mit Notenhals und einem Fähnchen	
Sechzehntel		ausgefüllter Notenkopf mit Notenhals und zwei Fähnchen	

Tabelle 2: Tabellarische Darstellung der Noten- und Pausenwerte von der ganzen Note bis zur Sechzehntel (angelehnt an eine Abb. aus [Zie88])

Zu jedem *Notenwert* gibt es auch ein entsprechendes Pausenzeichen. Bei der Notation von mehreren *Notenwerten*, die ein oder mehrerer Fähnchen besitzen, werden diese Noten durch sogenannte Balken miteinander verbunden.

Tempo

Notenwerte an sich geben keine Auskunft über die eigentliche Zeitdauer eines Tons. Eine solche Aussage ist erst durch die Festlegung eines *Tempos* möglich. Das *Tempo* wird durch die Anzahl der Viertelnoten pro Minute angegeben.

Takt/Taktarten

Takte sind die grundlegende zeitliche Struktur der Musik. Sie gruppieren bestimmte Notenwerte mit jeweils gleicher Zählzeit (gewissermaßen ein

Raster innerhalb des *Taktes*) und werden durch sogenannte *Taktstriche* voneinander getrennt. Die *Taktart* wird als mathematischer Bruch angegeben und nach dem Notenschlüssel und eventuellen Vorzeichen am Beginn der Notenzeile bzw. des *Taktes* notiert. Der Nenner gibt hierbei den rhythmischen Grundwert des *Taktes*, die sogenannte Takteinheit, an. Der Zähler gibt an, wie viele Grundwerte jeweils zu einem *Takt* gehören. Theoretisch können alle existierenden Notenwerte als rhythmischer Grundwert (z.B. Viertel) verwendet werden. „Viervierteltakt“ bedeutet also, dass die Summe der Notenwerte in einem Takt dem Wert von vier Vierteln entsprechen muss.

Punktierung

Wird eine Note punktiert, so ist die Dauer dieser Note um die Hälfte der ursprünglichen Dauer zu verlängern. Eine *punktierte* Viertel (siehe Abb. 8) hat also die Dauer von drei Achtelnoten. Gleiches gilt für die Punktierung von Pausen.

Abbildung 8: Punktierte Viertelnoten

Überbindung

Die *Überbindung* bewirkt, ähnlich wie die Punktierung, eine Verlängerung der Tondauer. Dabei wird eine Note mit gleicher Tonhöhe und beliebiger Tondauer durch einen *Haltebogen* an die ursprüngliche Note gebunden. Dies wird zum Beispiel genutzt, um Tondauern darzustellen, die durch Punktierung nicht darstellbar sind (z.B. fünf Achtel). Noten, die über einen Taktstrich hinaus gehalten werden sollen, können ebenfalls mit Hilfe von *Überbindung* notiert werden. Im Rahmen dieser Anwendung werden Haltebögen genutzt, um Tondauern darzustellen, die über einen Takt hinaus reichen würden.

Akkord

Ein *Akkord* bezeichnet den Zusammenklang/das gleichzeitige Spielen von drei oder mehr Tönen unterschiedlicher Tonhöhe. Die einzelnen Akkordtöne werden in der Notenschrift übereinander notiert.

Klaviatur

Als *Klaviatur* bezeichnet man alle Tasten eines Tasteninstruments, wie zum Beispiel eines Klaviers oder Keyboards.

Enharmonische Verwechslung

Die *enharmonische Verwechslung* lässt sich am einfachsten anhand der schwarzen Tasten einer Klaviatur erklären (siehe Abb. 9). Die Töne, die zu diesen Tasten gehören, tragen jeweils zwei Namen. Der Grund dafür ist, dass man sie entweder vom darüberliegenden Stammton (weiße Taste rechts der schwarzen) aus von oben herab oder vom darunterliegenden Stammton (weiße Taste links der schwarzen) benennen kann. Somit hat derselbe Ton nicht nur zwei Namen, sondern kann auch auf zwei Arten notiert werden

1. Notation des herabgesetzten, darüberliegenden Stammtons mit einem b als Vorzeichen

2. Notation des erhöhten, darunterliegenden Stammtons mit einem Kreuz als Vorzeichen

Die *enharmonische Verwechslung* beschränkt sich jedoch nicht nur auf die Töne der „schwarzen Tasten der Klaviatur“, sondern kann theoretisch auf alle Töne angewandt werden.

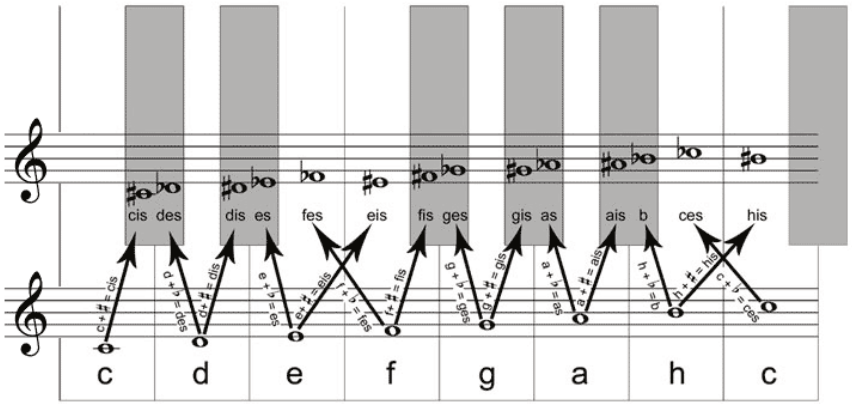


Abbildung 9: Darstellung der enharmonischen Verwechslung mit Hilfe einer Klaviatur[8]

[8]Quelle: http://www.lehrklaenge.de/HTML/Popup/EnharmonischeVerwechslungPopup.html

2.2 Verwendete Datenformate

Zur digitalen Verarbeitung von musikalischen Daten sind entsprechende Datenformate notwendig. In dieser Arbeit wurde zum einen das **MIDI-Format**, zum anderen das **MusicXML-Format** verwendet. Die wichtigsten Aspekte beider Formate werden im Folgenden näher betrachtet.

2.2.1 MIDI

Das *„Musical Instruments Digital Interface“* (deutsch:digitale Schnittstelle für Musikinstrumente) oder kurz *MIDI* [mida] ist ein System, das es musikalischen Instrumenten erlaubt miteinander zu kommunizieren und musikalische Daten in Echtzeit untereinander auszutauschen. *MIDI* stellt dazu eine für alle *MIDI*-fähigen Geräte einheitliche, musikbeschreibende Sprache in binärer Form bereit. Diese erlaubt es beispielsweise zu beschreiben, welche Note gerade gespielt wird und wie lange, schnell und laut sie gespielt wird. Ursprünglich wurde das Format für Keyboards entwickelt, ist inzwischen aber nicht mehr nur auf diese eine Instrumentengruppe beschränkt. Die Nachrichten können in sogenannten „Standard MIDI Files“ (kurz: SMF) abgespeichert werden. *MIDI* codiert die Musikdaten nicht als Tonaufnahme wie zum Beispiel mp3, sondern gibt nur Anweisungen wie die Musik gespielt werden soll. Somit ist die Sprache flexibel und gespeicherte Befehle bzw. Nachrichten können nachträglich korrigiert, verändert, gelöscht, beschleunigt oder verlangsamt werden.

Die drei Teile von MIDI Ursprünglich bestand *MIDI* nur aus einem Standard für die physische Verbindung der *MIDI*-Geräte und dem Nachrichtenformat zur Übertragung der Daten. Heute besteht *MIDI* offiziell aus drei verschiedenen Teilen, die häufig jeder für sich nur als „*MIDI*“ bezeichnet werden, obwohl sie unabhängige Teile mit unterschiedlichen Eigenschaften sind.

1. Das **MIDI Protokoll** bezeichnet die Spezifikation des *MIDI*-Nachrichten-Formats und ist der wahrscheinlich wichtigste der drei Teile. Das Protokoll beschreibt, wie die Nachrichten zur Codierung der Musikdaten auszusehen haben.

2. Die **MIDI Verbindungen**, die den physischen Kontakt zwischen den *MIDI*-Geräten herstellen. Anfangs wurden die Geräte ausschließlich über die *5-Pin DIN MIDI* verknüpft. Dieser Standard wurde 1983 zusammen mit dem Nachrichtenformat veröffentlicht und ist im Vergleich zu modernen Verbindungen eher langsam. Heute sind außerdem Anschlüsse über USB, FireWire, Ethernet, WiFi (LAN), Bluetooth oder Ähnlichem möglich. Die Geschwindigkeit der *MIDI*-Übertragungen hängen hierbei von der verwendeten Verbindung ab.

3. Das **Standard MIDI Files**-Dateiformat (kurz: SMF) und Varianten davon, die zur Speicherung von *MIDI*-Nachrichten verwendet werden können. Diese *MIDI*-Dateien (*.mid) können von allen geläufigen Computer-Plattformen abgespielt werden. Das Dateiformat kann sowohl für kommerzielle als auch nicht kommerzielle Zwecke uneingeschränkt verwendet werden.

MIDI-Nachrichten Wie schon erwähnt, ist *MIDI* ein binäres Format. Jede *MIDI*-Nachricht ist im Grunde nach dem gleichen Prinzip aufgebaut. Sie besteht aus einem sogenannten „Statusbyte“ gefolgt von für gewöhnlich ein bis zwei „Datenbytes“. Jedes Statusbyte beginnt mit einer „1“ und kann Werte zwischen ($80_{hex} - FF_{hex}$) annehmen, während Datenbytes jeweils mit einer „0“ beginnen und entsprechend Werte zwischen ($00_{hex} - 7F_{hex}$) annehmen. Beginnt eine Nachricht mit einem Datenbyte anstatt mit einem Statusbyte, so hat das letzte Statusbyte weiterhin seine Gültigkeit. Grob werden *MIDI*-Nachrichten in zwei Gruppen aufgeteilt:

1. „Channel Messages“: Sie beziehen sich je auf einen von insgesamt 16 möglichen *MIDI*-Kanälen. Dazu enthalten sie die entsprechende Kanalnummer in den vier niederwertigen Bits ihres Statusbyte

2. „System Messages“: Diese sind nicht kanalspezifisch und enthalten so auch keine Kanalnummer in ihrem Statusbyte.

Ein Überblick über alle verfügbaren Nachrichten wird auf der Homepage der MIDI Association [midb] bereitgestellt.

Im Folgenden werden die für dieses Projekt wichtigen Nachrichten-Typen kurz erläutert.

Channel Voice Messages „Channel Voice Messages“ zählen zu den Channel Messages und senden die Informationen, die für die musikalischen Ausführungen wichtig sind. Zu ihnen gehören die beiden für den Kontext dieser Arbeit wichtigsten Nachrichten „NoteOn“ und „NoteOff“. Dies sind die beiden Events, die das Drücken und Loslassen einer Taste signalisieren. Der Aufbau der beiden Nachrichten sieht wie folgt aus:

	Statusbyte		Datenbytes	
NoteOff	1000	nnnn	0kkk kkkk	0vvv vvvv
NoteOn	1001	nnnn	0kkk kkkk	0vvv vvvv

Tabelle 3: Aufbau der *MIDI*-Nachrichten NoteOn und NoteOff

Die ersten vier Bit des Statusbyte codieren den jeweiligen Befehl. Außerdem steht die Variable

- n für je ein Bit zur Codierung der Nummer des Kanals (0-15)

- k für je ein Bit zur Codierung der Nummer der Taste (siehe Abb. 25 im Anhang) (0-127)

- v für je ein Bit zur Codierung der „Velocity“, einer Maßzahl zur Beschreibung wie stark eine Taste angeschlagen wurde (0-127)

Der Velocity-Wert eines NoteOff-Events wird für gewöhnlich ignoriert. Ein NoteOn-Event mit Velocity-Wert 0 wird ebenfalls als NoteOff-Event interpretiert. Zu den Channel Voice Messages zählen außerdem „Aftertouch“-, „Pitch Bend“-, „Program Change“- und „Control Change“-Nachrichten, auf die aber hier nicht weiter eigegangen werden soll. Die Gruppe der Channel Messages beinhaltet außerdem die sogenannten „Channel Mode Messages“.

System Messages Als System Messages werden die „System Common Messages“, die sich an alle Empfänger im System richten, die „System Real Time Messages“, die genutzt werden, um zwei taktbasierte *MIDI*-Komponenten zu synchronisieren, und „System Exclusive Messages“ verstanden. Letztere werden genutzt, um Daten wie Patch-Parameter oder Beispieldaten zwischen *MIDI*-Geräten zu übertragen. Diese Nachrichten sind herstellerspezifisch und enthalten eine entsprechende Hersteller-ID gefolgt von beliebig vielen Datenbytes. Das Ende einer System Exclusive Message wird ebenfalls durch einen speziellen Befehl gekennzeichnet.

Auch an dieser Stelle wird auf weitere Ausführungen verzichtet, da die System Messages für diese Arbeit nicht relevant sind.

2.2.2 MusicXML

„MusicXML“ [mXM] bietet ein universelles Format zum Austausch und zur Verbreitung digitaler Noten in der gebräuchlichen, westlichen Musiknotation. Dabei wird die musikalische Information so wiedergegeben, dass sie sowohl von Notationsprogrammen und Sequenzern als auch von Musiklernsoftware oder Notendatenbanken eingesetzt werden kann. Inzwischen verwenden mehr als 200 verschiedene Programme *MusicXML*. Dies wird hauptsächlich dadurch möglich, dass das Dateiformat, wie der Name schon vermuten lässt, auf dem *XML*-Format (e*x*tensible *m*arkup *l*anguage) basiert. Die Dokumenttypdefinitionen (DTD) und W3C-XML-Schema-Definitionen (XSD) von *MusicXML* sind frei zugänglich und unter einer eigenen *MusicXML* Public License, die der W3C-Lizenz nachempfunden wurde, veröffentlicht.

Musikalische Daten im XML-Format Im Gegensatz zu hierarchisch gegliederten XML-Daten sind musikalische Daten eher gitterartig angeordnet. Sie müssen also entsprechend angepasst und umstrukturiert werden, um durch *MusicXML* dargestellt werden zu können. Grob gesagt besteht Musik aus mehreren Teilen („parts“), wobei jeder Teil für einen Spieler, ein Instrument, eine Stimme oder Ähnliches steht, und jeweils aus mehreren Takten („measures“). *MusicXML* bietet nun zwei Alternativen diesen Aufbau hierarchisch darzustellen. Diese werden durch zwei verschiedene DTDs mit entsprechendem eigenen Wurzelelement realisiert:

1. Wurzelelement **<score-timewise>**: Die Takte („measures“) stehen im Vordergrund und die einzelnen „parts“ des Musikstücks werden den einzelnen Takten untergeordnet angelegt. Für jeden Takt wird jeder „part“ je einmal notiert.

2. Wurzelelement **<score-partwise>**: Die musikalischen Teile („parts“) stehen im Vordergrund und die einzelnen Takte des Musikstücks werden jeweils dem entsprechenden Teil untergeordnet. Für jeden „part“ wird also jeder Takt je einmal notiert.

Beide Varianten sind äquivalent und können verlustfrei ineinander übersetzt werden. Im Rahmen dieser Arbeit wird mit nur einem einzigen „part“, dem MIDI-Keyboard, gearbeitet. Deshalb wurde die zweite Variante (<score-partwise>) genutzt. Es wird zu Beginn ein „part“ angelegt und anschließend die einzelnen Takte untergeordnet angehängt.

Die Elemente von MusicXML *MusicXML* unterscheidet zwei Hauptgruppen von XML-Elementen:

1. Elemente, die beschreiben, wie das Musikstück klingen soll: Diese Informationen sind auch in MIDI-Daten enthalten und können auch mehr oder weniger direkt aus den MIDI-Daten übernommen werden.

2. Elemente, die beschreiben, wie das Musikstück in der Notenschrift aussehen soll: Diese Informationen sind in den MIDI-Datensätzen nicht enthalten und finden hauptsächlich in Notensatzprogrammen oder ähnlichen Anwendungen, bei denen Notensätze erstellt werden, ihre Verwendung.

Da bei diesem Projekt nahezu ausschließlich mit den Daten gearbeitet wird, die die übertragenen MIDI-Nachrichten zur Verfügung stellen, werden sich die folgenden Ausführungen zu *MusicXML* ebenfalls auf diejenigen Elemente beschränken, die unter die erstgenannte Kategorie fallen.

Um die weiteren Elemente von *MusicXML* kennenzulernen, soll im Folgenden ein einfaches *MusicXML*-Dokument betrachtet werden. Die zum Verständnis nicht weiter relevanten, XML-spezifischen Definitionen am

Anfang des Dokuments (Zeilen 1-4) sollen an dieser Stelle ignoriert werden.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <!DOCTYPE score-partwise PUBLIC
3     "-//Recordare//DTD MusicXML 3.0 Partwise//EN"
4     "http://www.musicxml.org/dtds/partwise.dtd">
5  <score-partwise version="3.0">
6    <part-list>
7      <score-part id="P1">
8        <part-name>Music</part-name>
9      </score-part>
10   </part-list>
11   <part id="P1">
12     <measure number="1">
13       <attributes>
14         <divisions>1</divisions>
15         <key>
16           <fifths>0</fifths>
17         </key>
18         <time>
19           <beats>4</beats>
20           <beat-type>4</beat-type>
21         </time>
22         <clef>
23           <sign>G</sign>
24           <line>2</line>
25         </clef>
26       </attributes>
27     </measure>
28   </part>
29 </score-partwise>
```

Listing 1: "Hello World!" in *MusicXML*[9]

Abbildung 10 zeigt die Daten, die im XML-Dokument (siehe oben) in *MusicXML* codiert sind, in Notenschrift. Es handelt sich dabei um ein mittleres C (C4) im Violinschlüssel als ganze Note im Viervierteltakt.

Erst mit dem eigentlichen Wurzelelement <score-partwise ...>, welches schon hier kurz beschrieben worden ist, beginnen die für die Anwendung interessanten *MusicXML*-Informationen. Anschließend folgt eine Liste (<part-list>) der im Stück enthaltenen musikalischen „parts". Diese ist hier, wie auch bei der Verwendung im Projekt, minimal gehalten und weist nur

[9]Quelle: `http://www.musicxml.com/tutorial/hello-world/`

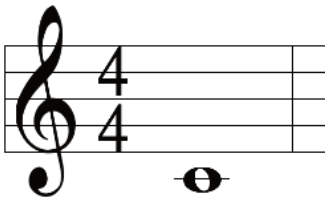


Abbildung 10: Mit der Anwendung gerendertes Ergebnis der oben stehenden XML-Datei

einen „part“ auf. Letztlich wird der „part“ geöffnet und die einzelnen Takte (<measure number=...>) notiert. Dabei erhält jeder Takt eine Nummer und setzt sich aus seinen Attributen (<attributes>) und einer (theoretisch beliebigen) Anzahl von Noten (<note>) mit Informationen zur Tonhöhe und -länge zusammen. Die Attribute beinhalten Angaben, die zur Interpretation der nachfolgenden Noten und musikalischen Daten notwendig sind. Dazu zählen unter anderem Auskünfte über die Taktart, Tonart oder den Notenschlüssel. Da es sich bei *MusicXML* um ein wohlgeformtes XML-Format handelt, werden zum Ende bzw. an den entsprechenden Stellen auch alle Elemente wieder geschlossen.

Attribute eines Taktes

- <divisions> gibt die Unterteilungen pro Viertelnote an. Das ist Grundlage für die Bestimmung/Darstellung der Dauer einer Note, die in Unterteilungen pro Viertel gemessen wird. Oft ist der Wert auf 24 festgelegt.

- <key> gibt die Tonart an. Hierbei steht <fifths> für die Anzahl der „b “s (negative Werte) und Kreuze (positive Werte). Dies bezieht sich auf die Position der Tonart im Quintenzirkel (siehe Abschnitt 2.1). „0“ steht für C-Dur ohne Vorzeichen, „2“ für D-Dur mit zwei Kreuzen, „-1“ für F-Dur mit einem b, etc. <mode> hingegen unterscheidet zwischen Dur („major“) und Moll („minor“).

- <time> beinhaltet die Taktangabe. Dazu wird in je einem Element der Zähler und der Nenner des Taktwertes angegeben.

- <clef> stellt den Notenschlüssel dar. Dafür wird das entsprechende Zeichen (G = Violinschlüssel, F = Bassschlüssel etc.) und die Basislinie, auf welcher der Schlüssel liegt, angegeben.

Die Noten des Taktes

- <pitch> stellt die Tonhöhe dar. Es wird der Sekundschritt (C, D, E, F, G, A, H) und die Oktave durch Angabe der „Nummer“ der Oktave (4

= Oktave, die beim mittlerem C startet) notiert. Optional kann durch das <alter>-Element ein Vorzeichen (2 = Doppelkreuz, 1 = Kreuz, 0 = neutral, -1 = b, -2 = Doppel-b usw.) dargestellt werden.

- <duration> enthält die Tondauer. Wie in der Musik üblich, wird diese durch einen Bruch dargestellt. Die Dauer steht hierbei immer in Relation zu den aktuellen <divisions>, welche die Anzahl der Unterteilungen einer Viertelnote bestimmen. Bei 24 Unterteilungen hat eine Viertelnote also eine Dauer von 24, eine Achtel eine Dauer von 12 und jede Achtel aus einer Triole[10] eine Dauer von 8.

- <type> spezifiziert, wie die Note notiert werden soll. Dies ist für gewöhnlich direkt aus der Dauer ableitbar. Es gibt jedoch Ausnahmen, beispielsweise bei der Notation von Jazzmusik oder speziellen Kurzschreibweisen, die in der Musik von J.S. Bach auftauchen.

- <chord/> wird genutzt, um Akkorde darzustellen. Dazu wird es für jede Note (abgesehen von der Ersten) im Akkord im zugehörigen Noten-Element direkt nach dem „Wurzelelement" der Note bzw. vor dem <pitch>-Element eingefügt.

- <tie type= ...> kann hinter dem <duration>-Element einer Note hinzugefügt werden und wird dazu verwendet gebundene Noten darzustellen. „type" kann hierbei die Werte „start" für die Note, bei der der Bindebogen beginnt, oder „stop" für die Note, bei der der Bindebogen endet, annehmen. Jedes <note>-Element kann bis zu zwei <tie>-Elemente besitzen, wenn die Note mit der vorhergehenden und der nachfolgenden gebunden ist.

Außerdem ist es durch das <rest/>-Element möglich eine Note als Pause zu deklarieren. In diesem Fall enthält das <note>-Element außer dieser Pausen-Deklaration nur noch eine Angabe zur Dauer (<duration>) der Pause.

Weitere Möglichkeiten Des Weiteren bietet *MusicXML* Möglichkeiten Liedtexte, Wiederholungen, Angaben dazu, wie die Daten dargestellt werden sollen, Akkord-Diagramme, Schlagzeugnotationen und vieles mehr zu verwenden.

2.3 Verwendete Bibliotheken

Im Verlauf des Implementierungsprozesses wurden nach und nach einige bereits existierende, externe Bibliotheken in das Projekt eingebunden. Bei

[10] Triole = Gruppe von drei Noten gleichen Notenwerts, die insgesamt nur den Zeitwert von zwei Noten ihrer Art haben.
Achteltriole = drei Achtel erklingen in der Zeit von zwei Achteln [Zie88]

der Auswahl der entsprechenden Programmbibliotheken ging es hauptsächlich um ihre Funktionalität, nicht zuletzt jedoch auch darum, dass sie jeweils leicht einzubinden, mit Windows kompatibel, möglichst kompakt, frei verfügbar und weitestgehend für die uneingeschränkte Nutzung freigegeben sind. Im Anschluss folgen jeweils kurze Beschreibungen der verwendeten Bibliotheken.

2.3.1 RtMidi

„*RtMidi*“ [rtM] stellt eine einfach API für die MIDI-Ein- und Ausgabe in Echtzeit bereit, die gemeinsam von Linux, Macintosh OS X und Windows genutzt werden kann. Dabei wird durch Kapselung der durch die betriebssystemspezifischen Bibliotheken für den Umgang mit MIDI-Signalen (ALSA & JACK (Linux), CoreMIDI & JACK (Macintosh) und die Multimedia Library (Windows)) durch *RtMidi* die Interaktion mit der systemeigenen MIDI-Hardware und Software für den Nutzer erleichtert.

RtMidi folgt einem objekt-orientierten C++ Design und besteht im Grunde nur aus einer einzigen Quell- und der dazugehörigen Header-Datei, die eine Reihe von C++ Klassen (*RtMidiIn*, *RtMidiOut* und einige API-spezifisch Klassen) beinhalten. Dies ermöglicht ein besonders leichtes Einbinden der Software in eigene Projekte.

Der Quellcode zu *RtMidi* ist auf GitHub verfügbar und wird unter einer MIT-Lizenz zur Nutzung freigegeben.

Verwendung von RtMidi Die MIDI-Ein- und Ausgabe ist auf zwei Klassen aufgeteilt, *RtMidiIn* und *RtMidiOut*, welche beide Subklassen der abstrakten Basisklasse *RtMidi* sind. Um die Bibliothek nutzen zu können, muss zunächst, da *RtMidi* nicht selbst instanziiert werden kann, je nach Verwendungszweck eine Instanz von *RtMidiIn* oder *RtMidiOut* erzeugt werden. Der Default-Konstruktor dieser beiden Klassen stellt hierbei, falls möglich, die notwendige Verbindung zum darunterliegenden MIDI-System her. Anschließend kann eine Verbindung zu einem verfügbaren MIDI-Port hergestellt werden. Dazu muss der Client zuerst alle erreichbaren MIDI-Ports abfragen und sich dann für einen entsprechenden Port entscheiden. Jede Instanz einer der Klassen erlaubt hierbei jeweils nur eine einzige MIDI-Port-Verbindung. Zusätzlich stellt *RtMidi* die Nutzung von „hot-pluggable“ und virtuellen MIDI-Geräten zur Verfügung, sodass das Verbinden von MIDI-Gräten, die zur Zeit der Instanziierung noch nicht angeschlossen waren, möglich ist. Außerdem bietet die Bibliothek die Funktion die von Linux- und Macintosh-Systeme bereitgestellte Möglichkeit virtuelle MIDI-Ports für die Ein- und Ausgabe einzurichten zu nutzen.

Im Folgenden wird die MIDI-Ein- und Ausgabe näher beschrieben.

MIDI-Ausgabe Die Klasse *RtMidiOut* stellt für die MIDI-Ausgabe eine Funktion bereit, um MIDI-Nachrichten über eine zuvor aufgebaute Verbindung zu senden. **Im Gegensatz zum MIDI-Format selbst** bietet die Bibliothek hierbei jedoch keine Timing-Funktion, was bedeutet, dass jede Nachricht der MIDI-Ausgabe ohne Verzögerung versendet wird. Auf weitere Ausführungen zur Ausgabe wird an dieser Stelle verzichtet, da sie in dieser Bachelorarbeit nicht zur Anwendung kamen.

MIDI-Eingabe Alle eingehenden MIDI-Signale werden von der Klasse *RtMidiIn* mit Hilfe einer internen Callback-Funktion vom angebundenen MIDI-Gerät bzw. dem entsprechenden Port empfangen und automatisch mit einem Zeitstempel, der die Delta-Zeiten zwischen dem letzten und dem aktuellen Signal in Sekunden in Form einer **double Gleitkommazahl** angibt, versehen. Anschließend werden die eingegangenen MIDI-Rohdaten in die einzelnen Bytes der Nachricht als Liste in Form eines *std::vector* unterteilt an den Benutzer übermittelt. Hierzu stellt die Klasse zwei Möglichkeiten bereit:

1. Die Nachrichten werden in einer Warteschlange gespeichert. Dies geschieht automatisch, wenn der Benutzer keine Callback-Funktion (siehe Punkt 2) registriert hat. Mit Hilfe der *RtMidiIn::getMessage()*-Funktion können die sich in der Warteschlange befindlichen MIDI-Nachrichten abgerufen werden. Die Funktion kopiert die Daten in einen *std::vector*, welcher vom Benutzer bereitgestellt werden muss. Ist die Warteschlange leer, so gibt die Funktion auch einen leeren Container zurück. Die verwendete Warteschlange hat grundsätzlich Platz für 1024 Nachrichten. Dieser Wert kann jedoch durch den Benutzer angepasst werden. Ist die Warteschlange voll, so werden alle im Anschluss eintreffenden Nachrichten verworfen, bis wieder Daten aus der Warteschlange entfernt worden sind.

2. Die empfangenen MIDI-Nachrichten werden direkt an eine benutzerdefinierte Callback-Funktion weitergeleitet, sobald der Nutzer eine solche bei *RtMidi* registriert hat. Wichtig ist, dass diese Registrierung der Callback-Funktion unmittelbar nach dem Öffnen des Ports vorgenommen wird, damit zuvor keine MIDI-Nachrichten in der Warteschlange abgelegt werden. Diese können dann nicht mehr durch die Callback-Funktion verarbeitet werden.

Fehlerbehandlung Die Fehlerbehandlung in *RtMidi* verwendet einen C++-Exceptionhandler, welcher in der Klasse *RtMidiError* implementiert ist. *RtMidiError* ist eine Klasse, die es erlaubt Fehler durch *RtMidiError::Type* abzufangen. Da viele *RtMidi*-Methoden *RtMidiErrors* werfen können, ist

es oft notwendig die Funktionsaufrufe in try/catch-Blöcke einzuschließen. Außerdem ist es möglich, eine benutzerspezifische Error-Callback-Funktion zu definieren, die beim Auftreten eines Fehlers aufgerufen wird.

Kompilieren Abschließend seien noch ein paar Hinweise zum Kompilieren von *RtMidi* für das entsprechende Betriebssystem mit der entsprechenden API gegeben. Hierzu müssen dem Compiler die zugehörigen Präprozessor-Definitionen und Bibliotheken bereitgestellt werden. Bei Windows-Systemen wird die Multimedia Library als MIDI-API verwendet. Für den Präprozessor muss _WINDOWS_MM_ definiert werden. Zudem muss die *winmm.lib* verknüpft werden.

2.3.2 TinyXML-2

„TinyXML-2“ [tin] bietet einen einfachen, kompakten und effizienten XML-Parser. Dieser ist in C++ geschrieben und ist somit problemlos auf allen C++-kompatiblen Systemen ausführbar. Er kann ohne Probleme in beliebige Programme und eigene Projekte integriert werden. Der Quellcode ist auf GitHub verfügbar und unter einer ZLib-Lizenz veröffentlicht.

Verwendung von TinyXML-2 *TinyXML-2* generiert beim Parsen der XML-Dokumente ein passendes Document Object Model (DOM). Die hierarchischen XML-Daten werden also in (C++-)Objekte umgewandelt. Das erstellte DOM kann dann gelesen, verändert und gespeichert werden. Außerdem kann mit dem Parser ein von Grund auf neues XML-Dokument auf Basis von C++-Objekten erstellt werden.

Die Ausgabe des flexiblen Parsers besteht aus korrektem und Standardkonformem XML. Sie ist UTF-8 Unicode codiert.

Das Hauptaugenmerk lag bei der Entwicklung von *TinyXML-2* darauf ein einfaches und schnell erlernbares System zu gestalten. Um den Parser in eigene Umgebungen einzubinden, muss ausschließlich eine einzige Quellcodedatei mit zugehöriger Header-Datei zum Projekt hinzugefügt werden.

Des Weiteren werden für die Verwendung von *TinyXML-2* weder Exceptions, noch RTTI [11] oder die STL [12] benötigt.

Im Gegensatz zu seinem Vorgänger (*TinyXML-1*) benötigt *TinyXML2* weniger Speicher-Allokationen, verwendet insgesamt weniger Speicher und arbeitet schneller. Außerdem wird, wie bereits erwähnt, die STL nicht mehr benötigt. Es verwendet modernes C++ und definiert einen eigenen Namespace („tinyxml2“).

[11] RTTI: **R**un**t**ime **T**ype **I**nformation, „Typinformationen zur Laufzeit“

[12] STL: generische C++ Bibliothek von Container-Klassen, Algorithmen und Iteratoren; stellt viele der Basisalgorithmen und Datenstrukturen der Informatik bereit [stl]

TinyXML-2 unterstützt hingegen keine DTDs [13] oder XLS [14]

Wichtige Klassen im TinyXML-2-DOM

- *XMLNode* ist die Basisklasse für nahezu alle Objekte im *TinyXML-2*-DOM. Ein *XMLNode* muss immer in einem XMLDocument enthalten sein. Er kann Geschwister-, Eltern- und Kinder-Knoten haben, die jeweils einzeln angesteuert werden können. Außerdem kann der spezifische Typ eines *XMLNodes* problemlos abgefragt und auf einen definierten Typ umgewandelt werden.

- *XMLDocument* vereint die gesamte Funktionalität des DOM. Diese Klasse ermöglicht das Laden, Speichern und Ausgeben der XML-Informationen. Alle Knoten innerhalb der XML-Daten sind einem Dokument zugewiesen und werden dort miteinander verknüpft. Ein jeder Kind-Knoten eines Dokuments (*XMLElement*, *XMLText*, etc.) kann nur durch einen entsprechenden, von *XMLDocument* bereitgestellten, Konstruktor erzeugt werden und ist somit auch direkt an das Dokument gebunden. Diese Knoten werden nach ihrer Erzeugung automatisch durch das Dokument verwaltet. Wenn ein *XMLDocument* gelöscht wird, so werden alle enthaltenen Knoten ebenfalls gelöscht.

- *XMLElement* ist eine einfache Container-Klasse. Sie besitzt einen Wert und einen Namen und kann außerdem andere Elemente, Texte, Kommentare etc. enthalten. Des Weiteren ist es möglich, einem *XMLElement* eine beliebige Anzahl an Attributen zuzuordnen, wobei jedes Attribut mit einem eindeutigen Namen versehen sein muss.

- Ein *XMLAttribute* ist nichts weiter als ein Paar bestehend aus einem Namen und einem zugehörigen Wert. Der Name sollte vorzugsweise eindeutig sein. Der Wert darf theoretisch von beliebigem Typ sein. Im Gegensatz zu den restlichen vorgestellten Klassen sind *XMLAttributes* keine *XMLNodes*.

2.3.3 FreeType2

„FreeType2“ [fre] ist eine Softwarebibliothek, die es ermöglicht, verschiedene Schriften mit Hilfe eines beliebigen Renderers darzustellen. Die Software ist in ANSI C geschrieben und unterstützt die meisten Vektor- und Bitmap-Font-Formate, wie etwa TTF-, CFF- oder PFR-Fonts. Bei der Gestaltung der Software wurde besonderer Wert auf Einfachheit, Effizienz, Anpassbarkeit und Portierbarkeit gelegt. Ziel von *FreeType2* ist es, eine leicht

[13] DTD: Document Type Definitions
[14] e**X**tensible **S**tylesheet **L**anguage

verwendbare und einheitliche Schnittstelle zu liefern, die Zugriff auf Font-Dateien verschiedener Formate gewährt. Weiterführende Funktionen, wie Textlayout oder grafische Verarbeitung, werden dabei nicht unterstützt. Um die Software verwenden zu können, müssen zunächst die entsprechenden Quellcode-Dateien heruntergeladen und die „ft2build.h“ sowie „FT_FREETYPE_H“ in das eigene Projekt inkludiert werden. Der Quellcode ist unter zwei Open-Source Lizenzen, einer BSD-ähnlichen FreeType-Lizenz sowie einer GNU General Public Lizenz auf der Homepage von *FreeType2* verfügbar und kann für alle Arten von Projekten verwendet werden.

Verwendung Die *FreeType2*-Bibliothek generiert aus den zum Beispiel aus Font-Dateien gegebenen Glyphenumrissen qualitativ hochwertige Glyphenbilder, welche dann beispielsweise wie einfache Texturen mit einem beliebigen Renderer dargestellt werden können. Es können entweder monochromatische Bitmaps oder kantengeglättete Pixmaps mit 256 Graustufen erzeugt werden.

Der modulare Aufbau der Bibliothek erlaubt zum einen die gewünschte Unterstützung vielfältiger Font- und Bild-Formate zum anderen gewährt er der Software eine hohe Flexibilität und Erweiterbarkeit. Die Module können entweder schon zur Compile-Zeit verknüpft oder aber erst zur Laufzeit geladen werden. Werden ausschließlich diejenigen Module, die für die aktuelle Anwendung benötigt werden, compiliert, so lässt sich außerdem die Größe des FreeType-Codes reduzieren.

Die C-Bibliothek sollte problemlos mit jedem C oder C++-Compiler kompatibel sein. Sollten keine speziellen Submodule für beispielsweise spezielle Font- oder Bildformate benutzt werden, besitzt *FreeType2* abgesehen von der ANSI C-Bibliothek keine weiteren externen Abhängigkeiten.

Dem Anwender ist es zudem möglich eine eigene Speicherverwaltung oder ein eigenes I/O-System, zum Beispiel für die Eingabe von Font-Daten zu verwenden. Effizientes Caching von Face-Instanzen oder Glyphenbildern wird zusätzlich durch ein Bibliothek-eigenes Cachingsystem erreicht.

Außerdem bietet *FreeType2* einfache Funktionen, um auf erweiterte Informationen, wie Glyphennamen oder grundlegende Kerningdaten, zuzugreifen.

FT_Library und FT_Face Die beiden für die im Rahmen dieses Projektes entwickelte Anwendung wichtigsten Elemente der *FreeType2*-Bibliothek sind:

- eine FT_Library-Instanz. Sie bildet das Wurzelelement für eine Menge von weiteren Objekten wie Fonts, Faces, Sizes etc. Sie enthält unter anderem einen eigenen Speichermanager. Bei der Erzeugung eines

FT_Library-Objektes werden alle *FreeType2* aktuell „bekannten“ Module in dieses Objekt geladen und können fortan verwendet werden. Sollte dabei ein Fehler auftreten gibt der Konstruktor, wie die meisten Funktionen der Bibliothek, einen Fehlercode zurück und setzt die FT_Library auf NULL.

- ein FT_Face. Dies modelliert eine gegebene Schrift in einem gegebenen Schriftstil (z.B. „Times New Roman Regular“ oder „Times New Roman Italic“). Bei Erzeugung des Objektes wird die angegebene Font-Datei geöffnet und falls möglich eine einzelne Schrift daraus geladen. Sind mehrere Schriftarten in der Datei enthalten, so kann über einen Index bestimmt werden, welche Schrift extrahiert werden soll. Alternativ kann auch eine Schrift aus dem Speicher der FT_Library geladen, oder mit Hilfe eines nutzereigenen I/O-Systems eingegeben werden. Anschließend kann mit Hilfe der FT_Face-Instanz auf die einzelnen Zeichen der Schrift und deren Bitmaps etc. zugegriffen werden und es können beispielsweise Pixelgrößen verändert oder Glyphenbilder geladen werden.

2.3.4 ImGui

Die *„ImGui“* [imG], auch bekannt unter dem Namen *„dear imgui“*, ist eine schnelle und portable C++-Bibliothek, die bei der Erstellung von grafischen Benutzerschnittstellen ihre Verwendung findet. Bei der Gestaltung der *ImGui* liegt der Fokus auf Einfachheit und Produktivität. Die Funktionen sind also auf die Wichtigsten beschränkt, werden jedoch auch fortwährend erweitert. Die *ImGui* eignet sich besonders gut für 3D-Echtzeitanwendungen, Vollbild-Anwendungen, eingebettete Anwendungen, Spiele oder Ähnliches. Die Bibliothek besteht aus ingesamt neun Dateien und ist in sich abgeschlossen. Dies bedeutet, dass sie keine weiteren externen Abhängigkeiten besitzt und es ausreicht die wenigen Quellcodedateien in das eigene Projekt zu kopieren, zu compilieren und in existierende Dateien zu inkludieren, um sie in das eigene Projekt einzubinden. Es ist kein spezieller Build-Prozess notwendig.

Die Übergabe der Maus- und Tastatureingaben sowie der gewünschten Einstellungen an *ImGui* ist Aufgabe des nutzereigenen Codes. Ist die Konfiguration der *ImGui* abgeschlossen, sind nur wenige Befehle notwendig, um diese zu verwenden.

Auch der Quellcode von *ImGui* ist auf GitHub verfügbar. Dort sind zusätzlich Beispielanwendungen veröffentlicht. *ImGui* wird unter einer MIT-Lizenz verbreitet.

Immediate Mode Rendering Die *ImGui* wurde nach dem Prinzip des Immediate Mode Rendering [imm] designed. Dahinter verbirgt sich ein Pro-

grammierungsstil für graphische Benutzeroberflächen. Im Gegensatz zum „Retained Mode“, bei dem zunächst aus den graphischen Primitiven eine Szene erstellt und beispielsweise als Liste von Objekten im Grafikspeicher abgelegt wird, bewirken beim „Immediate Mode“ die Aufrufe der entsprechenden Befehle, dass die Objekte ohne Zwischenspeicherung direkt auf den Bildschirm gerendert werden. Für jedes Frame müssen dann die Zeichenbefehle neu erstellt werden. Beim Immediate Mode Rendering bleibt der interne Zustand des darunterliegenden Rendersystems weitestgehend unberührt.

Verwendung der ImGui Die *ImGui* ist unabhängig vom verwendeten Renderer. Dies ist möglich, da sie VertexBuffer und einfache Befehlslisten ausgibt, die in der eigenen Anwendung mit Hilfe der verwendeten 3D-Renderpipeline (in diesem Falle OpenGL) gezeichnet werden können. Es wurde darauf geachtet, dass die Anzahl der Aufrufe der Zeichen-Befehle und der internen Zustandsänderungen möglichst gering gehalten wird. Die Verwendung des Immediate Mode Rendering macht es möglich, dass *ImGui*-Befehle überall im Code aufgerufen werden können. Die Bibliothek erlaubt es sowohl sehr aufwändige, aber auch sehr kurzlebige GUI-Werkzeuge zu erstellen. So ist es beispielsweise unter Verwendung des Edit&-Continue-Features moderner Compiler auch möglich Steuerelemente zum Bearbeiten von Variablen hinzuzufügen, während die Anwendung läuft und selbige kurze Zeit später schon wieder zu entfernen. *ImGui* eignet sich nicht nur zur Justierung von Variablen zur Laufzeit, sondern kann auch zur Verfolgung von laufenden Algorithmen durch einfache Textausgaben oder zur Erstellung eines Loggers, Profilers, Debuggers und Ähnlichem eingesetzt werden. Für die Text-Ein- und Ausgabe wird UTF-8 unterstützt. Die Möglichkeiten, die erstellte Benutzeroberfläche zu individualisieren sind beschränkt. Es können jedoch beispielsweise eigene Schriftarten im TrueType-Format geladen werden, oder die Farben, Größen, die Polsterung („Padding“) sowie die Rundungen der Grafikelemente mit entsprechenden Befehlen angepasst werden.

3 Implementierung

Im Anschluss an die Recherchephase (siehe Kapitel 2) folgte die eigentliche Implementierung der Anwendung, welche Thema dieses Kapitels sein soll. Der erste Teil soll zunächst das Endergebnis der Implementierung (den Prototypen) und seine einzelnen Funktionen vorstellen. Dies erleichtert das Verständnis der nachfolgenden Ausführungen der Implementierung. Daran schließt sich die Beschreibung des Implementierungsvorgangs, aus dem dieser Prototyp hervorgegangen ist, an. Hier wird erst ein kurzer Überblick (siehe Abschnitt 3.2) über die einzelnen Teile der Anwendung und de-

ren Verbindung untereinander gegeben. Danach werden die jeweiligen Abschnitte der Umsetzung (siehe Abschnitte 3.4, 3.5, 3.6 und 3.7) veranschaulicht. Außerdem wird sich eine kurze Erläuterung der projekteigenen Datenstrukturen und einiger kleiner „Datenbankstrukturen“ (siehe Abschnitt-3.1) anschließen. Bei den „Datenbanken“ handelt es sich zum Beispiel um *std::map*-Objekte, die Zuordnungen von MusicXML-Elementen zu einzelnen Zeichen (*GLchar*) für den Zugriff auf die Schrift-Texturen abspeichern.

3.1 Die Funktionen des Programms

Nachdem das MIDI-Keyboard, über das die Eingabe der Noten erfolgen soll, angeschlossen wurde, kann die Anwendung gestartet werden. Daraufhin öffnet sich am linken Bildschirmrand ein Fenster, welches die grafische Benutzeroberfläche zeigt (siehe Abb. 11).

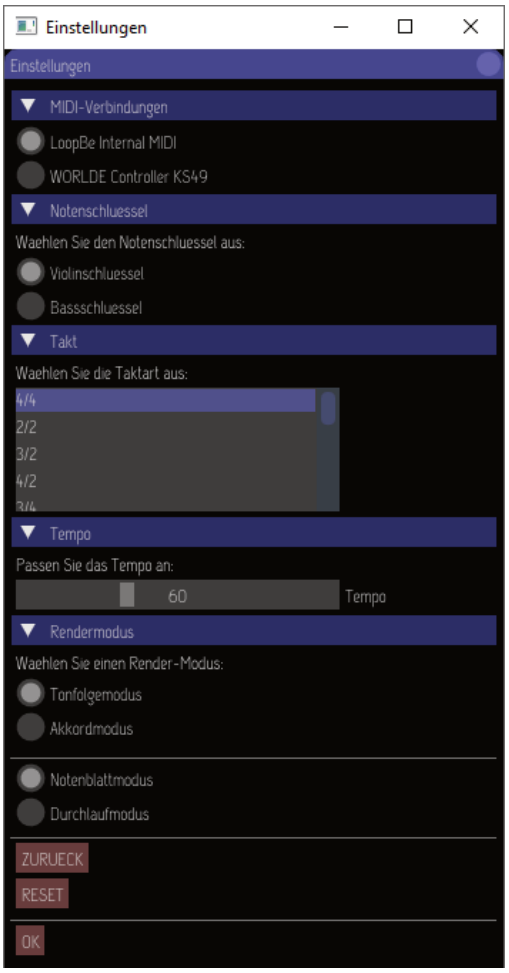


Abbildung 11: Grafische Benutzeroberfläche der Anwendung

Diese stellt verschiedene Einstellungsoptionen zur Verfügung. Einerseits können diejenigen Informationen, die für die Darstellung der Noten wichtig sind, jedoch nicht aus den MIDI-Daten hergeleitet werden können, eingegeben werden. Andererseits kann über die Oberfläche, falls mehrere MIDI-Geräte angeschlossen sind, der zu verwendende MIDI-Port gewählt werden oder zwischen den verschiedenen Darstellungsmodi gewechselt werden.

Im Folgenden werden die unterschiedlichen Einstellungsoptionen näher erläutert.

1. Auswahl der MIDI-Verbindung bzw. des MIDI-Geräts, welches zur Eingabe der MIDI-Daten verwendet werden soll (siehe Abb. 12):
   Unter dem Punkt „MIDI-Verbindungen“ werden alle der Anwendung bekannten MIDI-Ports aufgelistet. Wenn mehrere gültige Ports vorhanden sind, kann zwischen diesen gewählt werden. Wenn keine Verbindung angezeigt wird, muss die Anwendung geschlossen, entsprechende MIDI-Geräte angeschlossen und die Anwendung neugestartet werden. Die MIDI-Ports können jederzeit beliebig gewechselt werden.

Abbildung 12: Auswahl der MIDI-Verbindung bzw. des MIDI-Geräts

2. Auswahl des Notenschlüssels (siehe Abb. 13): Hier stehen der Violin- und der Bassschlüssel zur Verfügung. Es können beliebig viele weitere Notenschlüssel hinzugefügt werden. Standardmäßig ist der Violinschlüssel ausgewählt.

Abbildung 13: Auswahl des Notenschlüssels

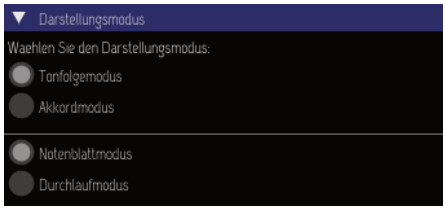
3. Auswahl der Taktart (siehe Abb. 14): Es stehen 16 verschiedene Taktarten zur Auswahl. Diese sind alle aus der verwendeten Schriftart entnommen. Auch diese Auswahl kann beliebig erweitert werden. Standardauswahl ist der Viervierteltakt.

Abbildung 14: Auswahl der Taktart

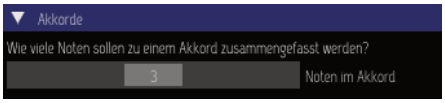
4. Einstellung des Tempos (siehe Abb. 15): Das Tempo wird durch die Anzahl der Viertelnoten pro Minute angegeben. Durch den Schieberegler kann ein beliebiger, ganzzahliger Wert zwischen 10 und 160 (Vierteln pro Minute) eingestellt werden. Standardmäßig ist der Wert auf 60 gesetzt.

Abbildung 15: Einstellung des Tempos

5. Auswahl des Darstellungsmodus (siehe Abb. 16): Der Darstellungsmodus gliedert sich in zwei Teile. Aus beiden Teilen muss je ein Modus gewählt werden. Der erste Teil unterscheidet zwischen dem Tonfolgemodus und dem Akkordmodus, der zweite zwischen dem Notenblattmodus und dem Durchlaufmodus. Ist der Akkordmodus eingestellt, so kann durch einen Schieberegler (siehe Abb. 16 (b)) die Anzahl der Noten, die jeweils einen Akkord bilden sollen, angepasst werden. Es sind zwischen einer und fünf Noten möglich. Die einzelnen Modi werden im Anschluss kurz erläutert. Standardauswahl sind der Tonfolgemodus und der Notenblattmodus.



(a) Wahl des Darstellungsmodus (Tonfolgemodus ausgewählt)



(b) Schieberegler zur Einstellung der Anzahl der Akkordnoten

Abbildung 16: Auswahl der Darstellungsmodi

Tonfolgemodus Im *Tonfolgemodus* (siehe Abb. 17) werden die Noten als Tonfolge im Notensystem mit dem eingegebenen Notenschlüssel und der eingegebenen Taktart dargestellt. Die Töne werden angezeigt, sobald eine zuvor gedrückte Taste auf der Klaviatur wieder losgelassen wird. Die Tonlänge wird automatisch dem Tempo und der Taktart angepasst. Außerdem werden im *Tonfolgemodus* einzelne Takte durch Taktstriche unterteilt. Sobald ein Takt gemäß der eingestellten Taktart vollständig ist, wird ein Taktstrich gezogen. Das gleichzeitige Betätigen von mehreren Tasten des Keyboards führt auch im *Tonfolgemodus* zur Erzeugung eines Akkords. Die Länge der Töne in diesen Akkorden wird durch die Länge des Tons bestimmt, dessen zugehörige Taste auf der Klaviatur als erstes losgelassen wird. Der Akkord wird ebenfalls erst nach dem Loslassen dieser Taste angezeigt.

Akkordmodus Im *Akkordmodus* (siehe Abb. 18 und Abb. 19) werden ausschließlich möglich Akkorde eingegeben. Die einzelnen Töne der Akkorde werden hierbei jedoch im Gegensatz zum Tonfolgemodus nacheinander eingegeben, das heißt es ist keine gleichzeitige Betätigung mehrerer Tasten der Klaviatur nötig. Entsprechend wird jeder Ton des Akkordes auch dargestellt, sobald seine Taste losgelassen wurde und nicht erst, wenn alle Töne eingegeben sind. Hierzu muss bei Auswahl des *Akkordmodus* wie bereits erwänt, die Anzahl der Noten, die zu einem Akkord zusammengefasst werden sollen, eingestellt werden. Die Länge der eingegebenen Töne wird in diesem Modus ignoriert und es werden ausschließlich ganze Noten angezeigt. Aus diesem Grund steht auch am Beginn der Notenzeilen auf dem Bildschirm im *Akkordmodus* nur ein Notenschlüssel und **keine** Taktart. Außerdem werden keine Taktstriche gezogen.

Im *Notenblattmodus* (siehe Abb. 20) werden vier Notenzeilen im Anzeigefenster dargestellt. Wird das Ende einer Notenzeile erreicht, erzeugt die Anwendung automatisch einen Zeilenumbruch und fährt mit der Darstellung der Noten in der nächsten Notenzeile fort.

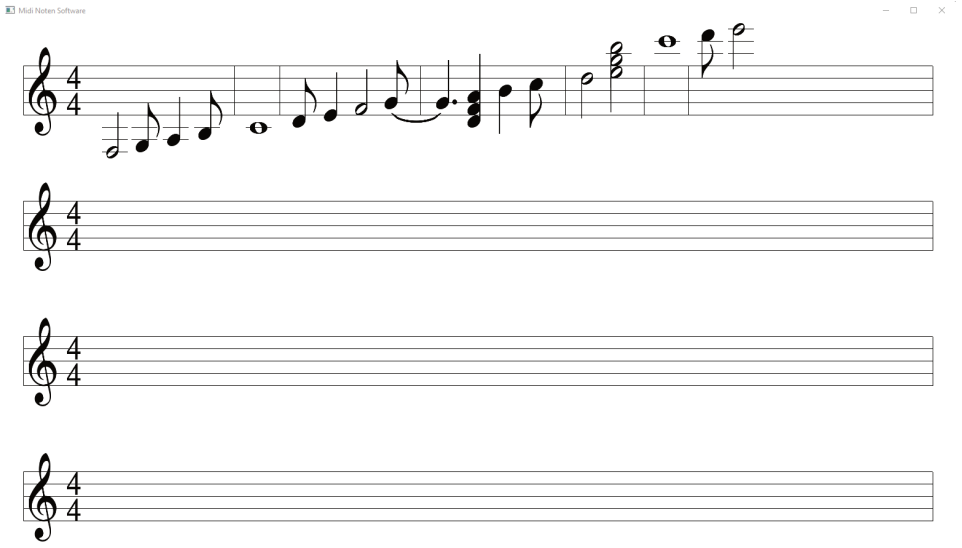


Abbildung 17: Der Tonfolgemodus

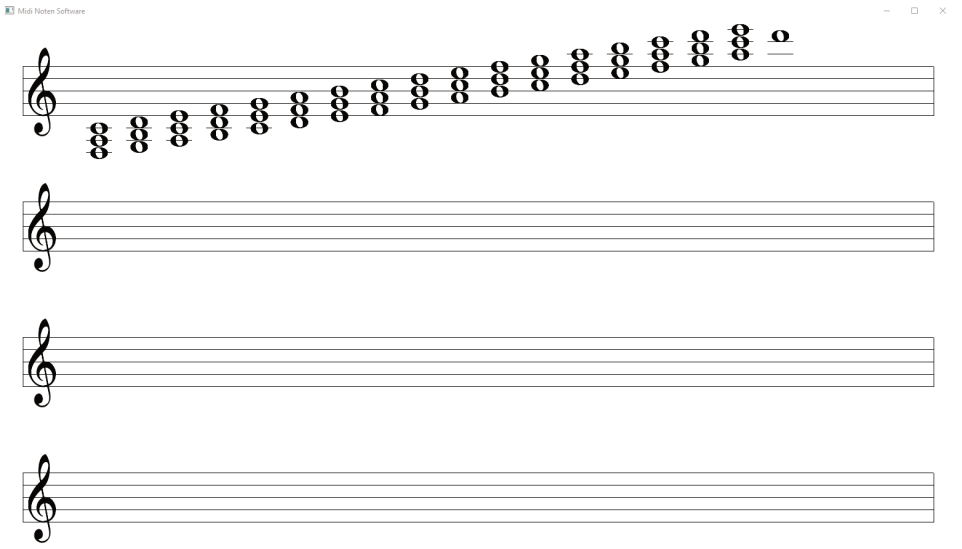


Abbildung 18: Akkordmodus mit je drei Noten pro Akkord

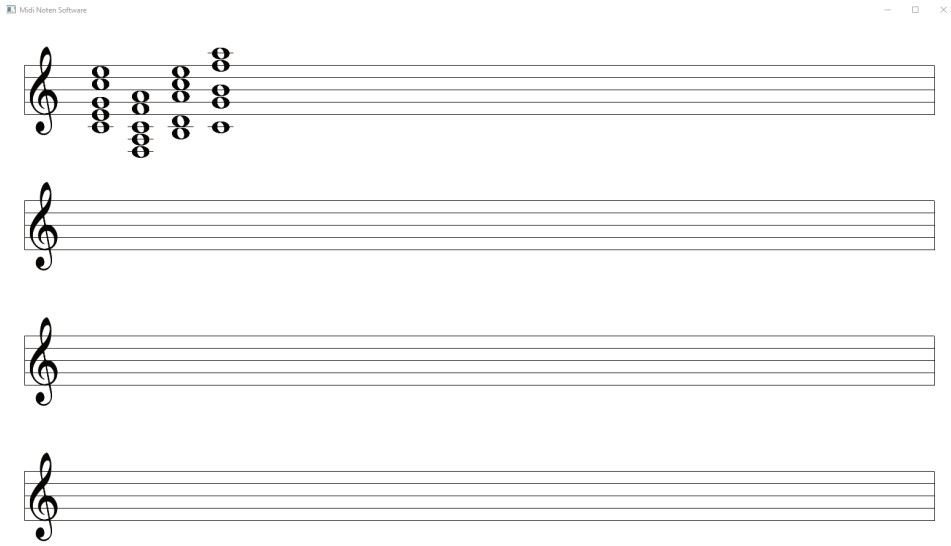


Abbildung 19: Akkordmodus mit je fünf Noten pro Akkord

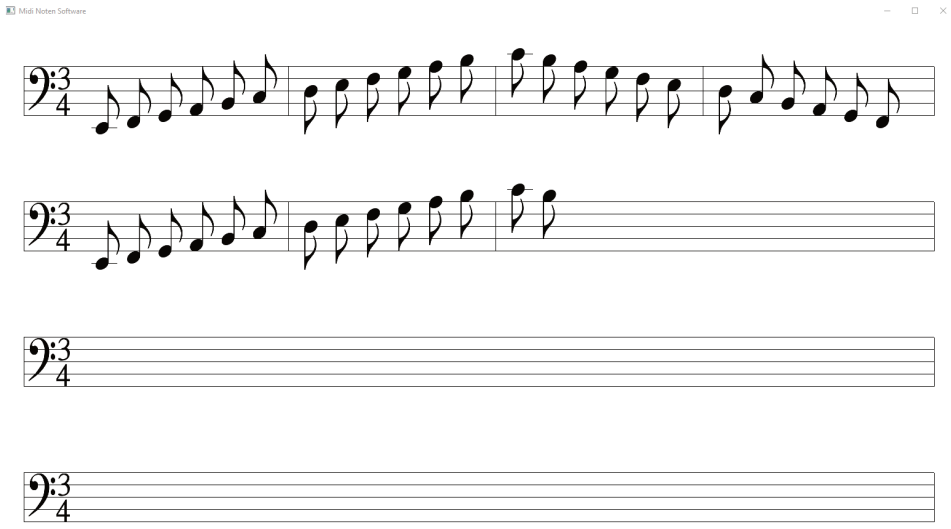


Abbildung 20: Der Notenblattmodus

Durchlaufmodus Im *Durchlaufmodus* (siehe Abb. 21) wird nur eine einzige Notenzeile im Anzeigefenster[15] dargestellt. Wird das Ende der Notenzeile erreicht, werden die angezeigten Noten nach links gerückt, sodass am Ende der Zeile Platz für weitere Noten entsteht. Die Noten, die am Anfang der Zeile stehen, werden hierbei nach links aus der Anzeige „herausgeschoben“ und fortan nicht mehr angezeigt.

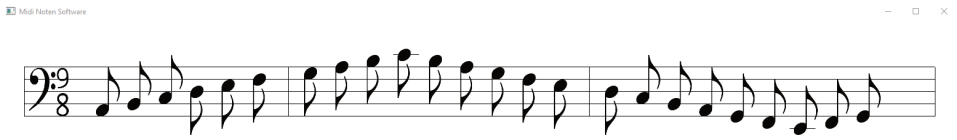


Abbildung 21: Der Durchlaufmodus

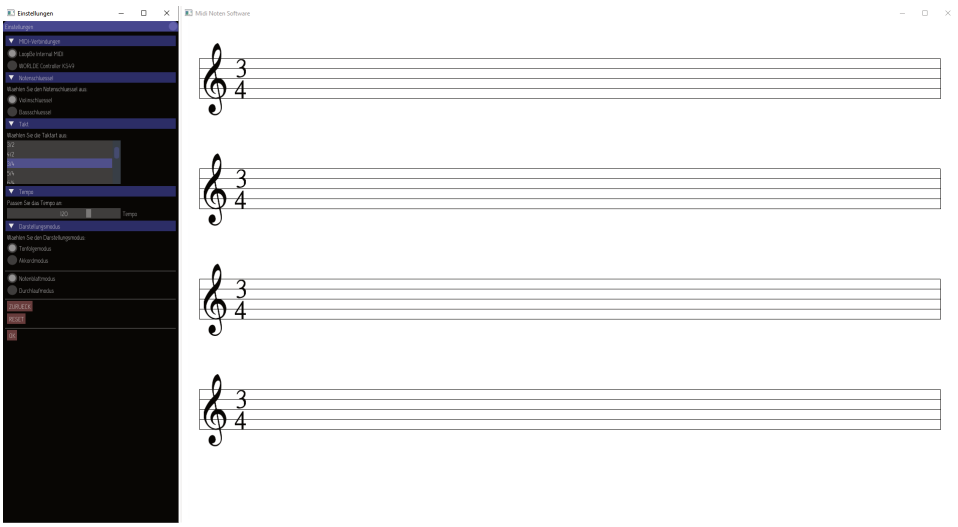


Abbildung 22: Bildschirmausgabe nach Bestätigung der initialen Einstellungen durch „OK“-Knopf

Abschließend werden die getätigten Einstellungen durch Drücken des „OK“-Knopfes bestätigt. Nun erscheint im Bildschirm rechts neben dem Fenster mit der grafischen Benutzerschnittstelle ein weiteres Fenster mit Notenlinien und dem gewählten Notenschlüssel und Takt (siehe Abb. 22). Damit ist die Konfiguration abgeschlossen und es können ab sofort Noten mit Hilfe des MIDI-Keyboards eingegeben werden. Diese erscheinen entsprechend im rechten Fenster.
Je nachdem welcher Notenschlüssel gewählt wurde, können unterschiedliche Notenbereiche dargestellt werden. Der Violinschlüssel umfasst hier

[15] Fenstergröße ist dieselbe wie beim Tonfolgemodus

die Töne zwischen f und e''', der Bassschlüssel die Töne zwischen A, und g'. Wenn Töne eingeben werden, die außerhalb des momentan gültigen Bereichs liegen, so werden diese ignoriert.

Nachträgliche Änderungen der Einstellungen müssen ebenfalls durch Betätigen des „OK"-Knopfes bestätigt werden. Einige Veränderungen wie beispielsweise die nachträglichen Änderung des Notenschlüssels oder der Taktart bewirken, dass die Anzeige der Noten zurückgesetzt wird, da zur Zeit noch keine Funktion zur Aktualisierung der gesamten XML-Datei[16] zur Verfügung steht.

Die beiden Knöpfe „ZURUECK" und „RESET", die während der Konfiguration noch keine Funktion hatten, können jetzt verwendet werden. Der „ZURUECK"-Knopf löscht die **zuletzt** eingegebene Note. Wurde zuletzt ein Akkord eingegeben (gilt nur für den Tonfolgemodus), wird dieser als Ganzes gelöscht. Wird eine Note im Durchlaufmodus entfernt und somit eine Stelle in der Anzeige frei, rücken die Noten von links aus wieder nach. Das bedeutet, dass eine Note, die zuvor nach links aus der Anzeige „herausgeschoben" wurde, nun wieder abgebildet wird. Der „RESET"-Knopf setzt die gesamte Anzeige zurück, das heißt er löscht **alle** eingegebenen Noten. Durch das Schließen von einem der beiden Fenster wird die Anwendung beendet.

Die akustische Wiedergabe der eingegebenen Töne wird derweil noch nicht unterstützt. Die Balkennotation von kleineren Notenwerten (Achtel und kleiner) ist ebenfalls nicht möglich.

3.2 Überblick

Die Implementation des Prototypen gliederte sich in vier verschiedene Abschnitte. Zunächst wurde die Eingabe (siehe Abschnitt 3.4) der Noten durch das MIDI-Keyboard umgesetzt. Dieser Teil beinhaltet den „Reader" (siehe Abschnitt 3.4.1) und den „Parser" (siehe Abschnitt 3.4.2). Der Reader liest die MIDI-Daten von dem MIDI-Port aus und gibt diese an den Parser weiter. Der Parser nimmt die MIDI-Daten dann vom Reader entgegen und wandelt sie in ein, von den nachfolgenden Elementen der Verarbeitungspipeline der Anwendung lesbares, Format (MusicXML) um. Im zweiten Abschnitt wurde die visuelle Ausgabe (siehe Abschnitt 3.5) der eingegebenen Daten implementiert. Der Ausgabeteil besteht hauptsächlich

[16]Diese Funktion müsste beispielsweise die existierenden Takte neu berechnen und die Noten in der Datei gegebenenfalls in andere (vorhergehende oder nachfolgende) Takte verschieben oder die Datei in Bezug auf die „Gültigkeit" der Töne im gegenwärtigen Notenschlüssel aktualisieren und ungültige Töne löschen oder für die spätere Wiederverwendung zwischenspeichern.

aus dem „Renderer“ (siehe Abschnitt 3.5.1), welcher die vom Parser formatierten Eingabedaten einliest und in eine visuelle Wiedergabe auf dem Bildschirm umsetzt. Der dritte Abschnitt bestand dann darin die Verknüpfung der beiden zunächst unabhängigen Komponenten für die Ein- und Ausgaben herzustellen. Dies geschah durch die Implementierung des „XMLManagers“ (siehe Abschnitt 3.6.1). Der XMLManager verwaltet die erforderlichen XML-Dateien und stellt Methoden bereit, um in diese zu schreiben und sie zu verändern oder sie zu lesen. Im letzten Abschnitt wurde der Anwendung noch eine grafische Benutzerschnittstelle (siehe Abschnitt 3.7) hinzugefügt, die beispielsweise die Nutzereingabe von Parametern, welche nicht aus den MIDI-Daten hergeleitet werden können, erleichtert. Der Arbeitsablauf der Anwendung wird in Abbildung 23 schematisch dargestellt. Diese Abbildung ist in größerer Ansicht noch einmal im Anhang zu finden 24.

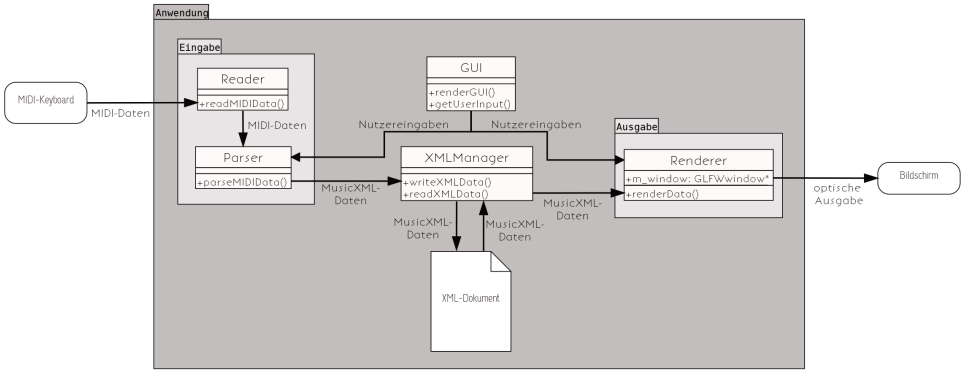


Abbildung 23: Verarbeitungspipeline der Anwendung

3.3 Projekteigene Daten- und Datenbankstrukturen

Im Verlauf der Implementierungsphase erwies es sich als sinnvoll nach und nach einige Datenstrukturen in Form von *C++-structs* zu definieren, um die Verwaltung der benötigten Daten und den Zugriff darauf einfacher zu gestalten. Sie werden zum Beipiel für das Zwischenspeichern von MIDI-Nachrichten oder auch das Ablegen von Notendaten verwendet. Sie bestehen jeweils nur aus einer Header-Datei und stellen (bis auf zwei Ausnahmen) keinerlei Funktionen bereit. Gleichermaßen wurden in einigen speziellen Header-Dateien globale Konstanten angelegt. Diese fungieren als eine Art Datenbank und speichern beispielsweise die feste Zuordnung von MIDI-Notennummern zu Notennamen, Oktaven und Versetzungszeichen ab. Beide Komponenten werden nun vorgestellt.

3.3.1 Die Datenstrukturen

MidiEvent *MidiEvents* werden ausschließlich von der Parser-Klasse verwendet. Sie dienen dazu die gerade geparste MIDI-Nachricht aufgeschlüsselt zwischenzuspeichern. Jedes *MidiEvent* besitzt einen „StatusType“ (siehe Abschnitt 3.3.2). Dieser übernimmt die Speicherung des Statusbyte der MIDI-Nachricht mit Hilfe eines *Enum*-Typs. Zusätzlich verfügt ein *MidiEvent* über zwei Integer-Werte, je einen für die MIDI-Kanalnummer der Nachricht und für die „Velocity“ im Falle einer *NoteOn*- oder *NoteOff*-Nachricht. Darüber hinaus dient ein Double-Wert zur Speicherung des Zeitstempels der Nachricht und ein *std::string* als ID für die Identifikation von zusammengehörigen *NoteOn*- und *NoteOff*-Events. Außerdem stellt *MidiEvent* eine Funktion bereit, die zwei Events auf Gleichheit testen kann und einen entsprechenden Bool-Wert zurückgibt.

Measure Die *Measure*-Struktur wird sowohl vom Renderer als auch vom XMLManager genutzt um Taktstrukturen zu verwalten. Dazu kennt jeder *Measure* seine „divisions“ (Integer), also die Anzahl der Unterteilungen einer Viertelnote (siehe Abschnitt 2.2.2), seine Ton- und Taktart (Integer und Float), seinen Notenschlüssel (siehe Abschnitt 3.3.1) und eine Liste von zum Takt gehörenden Noten (siehe Abschnitt 3.3.1) (*std::vector*). Außerdem weiß jeder Takt, ob er Attribute (im Sinne der MusicXML-Attribute des Takts) besitzt (boolescher Wert).

Clef *Clef* fungiert als Datenstruktur für Notenschlüssel. Jeder *Clef* verfügt über ein Zeichen (*std::string*) und eine Grundlinie, auf der der Notenschlüssel liegt. Für den Violinschlüssel wäre dies das „G“ und die Linie zwei im Notensystem.

Note Wie der Name schon sagt dient die *Note*-Struktur dazu die einzelnen Daten einer Note zusammenzufassen. *Note* speichert hierzu neben den Informationen aus den MusicXML-Daten, ob und wie viele Hilfslinien (Integer) die Note benötigt, ob sie an eine andere gebunden ist und ob die Note einem Akkord zugeordnet ist (jeweils ein boolscher Wert). Zu den MusicXML-Daten zählen der Namen des Stammtons (Character), der Index der Oktave (Integer), das Vorzeichen (Integer; entsprechend der Codierung, die auch in MusicXML verwendet wird) sowie die Dauer und der Typ der Note. Des Weiteren stellt *Note* eine Methode bereit, die zwei *Note*-Instanzen auf Gleichheit prüfen kann.

3.3.2 Die Datenbanken

Status *Status* dient zur Umwandlung der hexadezimal codierten Statusbytes der MIDI-Nachrichten in ein einfacher verwendbares und für den

Menschen leichter lesbares Format. Es besteht aus zwei Teilen. Zum einen aus einer *Enumeration*, welche die verschiedenen Typen von Statusbytes von MIDI-Nachrichten (NoteOn, NoteOff etc.) auflistet, zum anderen aus einer `std::map`, welche jedem hexadezimalen Statusbyte einen solchen Typen zuweist.

Pitch *Pitch* fungiert ebenfalls als Vermittler zwischen den hexadezimalen MIDI-Daten und dem Daten-Format, welches für MusicXML benötigt wird. Hauptbestandteil ist ein 128-stelliges Array. Diese weist der, im ersten Datenbyte der NoteOn- und NoteOff-Nachrichten codierten, Tonhöhe eine Note (siehe Abschnitt 3.3.1) mit Stammton, Oktave und Vorzeichen zu. MIDI deckt die Oktavbereich von „‚C bis g"''' ab.

CharacterData Die *CharacterData* wird ausschließlich vom Renderer für die Handhabung der Schrifttexturen genutzt. Sie besteht aus drei Teilen.

„charMap" Für jede Schrift, die in der Anwendung zum Einsatz kommen soll, wird eine *charMap*, also eine Zeichentabelle benötigt. Sie besteht aus einer *std::map*. Diese bildet die Begriffe zur Beschreibung der Notentypen, Pausentypen, Notenschlüsseln, Taktarten etc. aus dem MusicXML-Format auf diejenigen Zeichen (*GLchar*) ab, die beim Rendern für den Zugriff auf die passende Schrift-Textur benötigt werden.

„noteLine" Jeder von der Anwendung unterstützte Notenschlüssel muss entsprechende Einträge in diese *std::map* besitzen. Hier wird für die jeweils im Notenschlüssel gültigen Tonbereiche der vertikale Versatz des Tons in Relation zur Basislinie[17] bzw. zur Basisnote[18] des Notensystems angegeben.

„specificScale" Hiermit kann jedem Zeichen (GLchar) und damit jeder Schrift-Textur eine spezifische Skalierung in Form eines float-Wertes zugeordnet werden, welche dann beim Rendervorgang berücksichtigt wird. Dies wurde aufgrund der teils starken Größenunterschiede innerhalb der aktuell genutzten Schriftart [19] notwendig.

RenderMode *RenderMode* bietet zwei kleine *Enum*-Typen an, die die Unterscheidung der verschiedenen Rendermodi erleichtern. „ModeA" differenziert zwischen dem Tonfolgemodus und dem Akkordmodus, während „ModeB" den Notenblattmodus vom Durchlaufmodus trennt.

[17] erste/unterste Linie im Notensystem

[18] Hier die Note, welche zwischen der ersten und zweiten Linie im Notensystem liegt; Violinschlüssel: f', Bassschlüssel: A

[19] MusiSync.tff [fon]

3.4 Die Eingabe

Zum Bereich „Eingabe“ zählen die Reader- und die Parser-Klasse. Diese kümmern sich darum, dass die MIDI-Daten vom MIDI-Anschluss ausgelesen werden und in das MusicXML-Format überführt werden.

3.4.1 Der Reader

Die *Reader*-Klasse ist eine Klasse, die für das Einlesen der MIDI-Daten vom Keyboard zuständig ist. Für den Zugriff auf den MIDI-Port, an dem das Keyboard angeschlossen ist, und den Zugriff auf die MIDI-Daten selbst verwendet sie die Bibliothek RtMidi (siehe Abschnitt 2.3.1). Dabei nutzt der *Reader* die MIDI-Nachrichten-Warteschlange, die RtMidi automatisch bereitstellt, wenn keine benutzereigene Callback-Funktion definiert ist, die die eingehenden Nachrichten verarbeitet.

Wird ein neues *Reader*-Objekt angelegt, so erzeugt der Konstruktor der Klasse als erstes eine Instanz der Klasse „RtMidiIn“, welche das Einlesen der MIDI-Daten und die eintreffenden MIDI-Nachrichten verwaltet. Danach werden mit Hilfe dieser RtMidiIn-Instanz alle verfügbaren MIDI-Anschlüsse abgefragt und in einem *std::vector* gespeichert. Ist kein MIDI-Anschluss erreichbar, so bleibt die Liste der Anschlüsse (der *std::vector*) leer. Zwischen verfügbaren MIDI-Anschlüssen kann zur Laufzeit gewechselt werden, wobei immer nur ein einziger Anschluss gleichzeitig aktiv sein kann.

Bevor der *Reader* mit seiner Arbeit beginnen kann, muss einer der gültigen MIDI-Anschluss geöffnet werden. Dies geschieht in der main-Methode. Die Funktion für das Öffnen eines Anschlusses bekommt die Nummer bzw. den Index des zu öffnenden Anschlusses als Parameter übergeben. Ist das RtMidiIn-Objekt noch mit keinem Anschluss verbunden, wird direkt eine neue Verbindung zum entsprechenden Anschluss geöffnet. Ist schon ein Verbindung vorhanden und ist der übergebene Anschluss ein anderer als der aktuell verbundene Anschluss, so wird die alte Verbindung zunächst geschlossen und erst dann eine neue Verbindung zum neuen Anschluss eingerichtet.

Nun kann der *Reader* mit dem Einlesen der Daten beginnen. Dies geschieht mit Hilfe der *read()*-Funktion, welche in jedem Durchlauf der Renderschleife aufgerufen wird. Die *read()*-Funktion fragt dann die aktuelle Nachrichten-Warteschlange des RtMidiIn-Objektes ab und speichert sie in einer Liste (*std::vector*), die der Funktion als Parameter übergeben wird, ab. Des Weiteren gibt sie den Zeitstempel der zuletzt eingegebenen Nachricht in Form eines *double*-Werts zurück.

3.4.2 Der Parser

Die *Parser*-Klasse ist für die Weiterverarbeitung der vom Reader eingelesenen MIDI-Daten zuständig. Er erhält die MIDI-Daten als Eingabe für den Parsevorgang, wandelt diese in das MusicXML-Format um und schreibt die MusicXML-Daten mit Hilfe des XMLManagers in eine Datei. Dabei übersetzt die Klasse beim Parsen nicht einfach nur die Tonhöhe inklusive Vorzeichen, sondern bestimmt zusätzlich die Tondauer, fasst Akkorde zusammen, legt Takte fest und verwaltet den aktuellen Notenschlüssel sowie die Taktart (siehe Abschnitt 2.1).

Der Konstruktor Aufgabe des Konstruktors der *Parser*-Klasse ist es nur die Standardwerte der Membervariablen zu setzen und das *Parser*-Objekt über eine Referenz mit einer Instanz des XMLManagers zu verknüpfen.

Sofort nach Erzeugung eines neuen *Parser*-Objektes sollte sichergestellt werden, dass die vom *Parser* benötigten Parameter gesetzt werden. Dazu zählen hauptsächlich die Größen, die über die Benutzeroberfläche eingegeben werden müssen wie Taktinformationen (Notenschlüssel, Takt- und Tonart), das Tempo oder auch der aktuelle Darstellungsmodus. Die Taktinformationen werden dann direkt als Attribute des ersten Takts in die MusicXML-Datei, welche bereits durch den XMLManager angelegt wurde, eingetragen.

Für die Übergabe der MIDI-Daten vom Reader an den *Parser* werden diese vom Reader in einen *std::vector* (Membervariable des *Parsers*) geschrieben. Außerdem muss der Zeitstempel der MIDI-Nachricht an den *Parser* übergeben werden. Der Parsevorgang wird dann durch den Aufruf der *parse*-Methode gestartet.

Die *parse*-Methode setzt als Erstes den aktuellen Status des *Parser*-Objekts. Dieser entspricht dem im Statusbyte der MIDI-Nachricht codierten Nachrichtentyp (z.B. NoteOn oder NoteOff). Daraufhin wird zur vereinfachten Verwaltung der MIDI-Daten ein MidiEvent-Objekt (siehe Abschnitt 3.3.1) angelegt, welches die aufgeschlüsselten Datenbytes [20] der Nachricht speichert. Danach wird eine für jeden Nachrichtentyp spezifische *parse*-Methode aufgerufen. Hier werden zur Zeit nur NoteOn- und NoteOff-Nachrichten unterstützt.

Parsen von NoteOn-Nachrichten Für jede eintreffende NoteOn-Nachricht wird als Erstes bestimmt, ob der in ihr codierte Ton innerhalb des gerade aktiven Notenschlüssels gültig ist oder ob er außerhalb des darstellbaren (Oktav-)Bereichs liegt. Für den Violinschlüssel umfasst dieser Bereich die Töne von f bis e''', für den Bassschlüssel die Töne von ‚A bis g'. Ungültige Töne werden ab sofort ignoriert. Bei gültigen Tönen wird das anfangs

[20] aufgeteilt nach Note/Tonhöhe und „Velocity"(Anschlagsstärke)

erstellte MidiEvent in einer Liste (*std::vector*) abgespeichert, welche die abzuarbeitenden Events verwaltet. Falls zwei NoteOn-Nachrichten unmittelbar aufeinanderfolgen wird außerdem an dieser Stelle ein neuer Akkord gestartet und die zum Akkord gehörenden Events in einer weiteren Liste (*std::vector*) abgelegt.

Parsen von NoteOff-Nachrichten Hinter dem Parsevorgang einer NoteOff-Nachricht verbirgt sich ein Großteil der Funktionalität des *Parsers*. Wird eine NoteOff-Nachricht vom Reader an den *Parser* weitergeleitet, so sucht dieser erst das dazugehörige NoteOn-Event aus der Liste von gespeicherten Events heraus. Dazu wird die ID des NoteOff-Events mit den jeweiligen IDs der Events aus der Liste abgeglichen. Die MidiEvent-ID besteht hierbei aus dem Namen des Stammtons, der Oktave und der Vorzeichencodierung der entsprechenden Note. Anschließend wird anhand des Zeitstempels der NoteOff-Nachricht die Dauer des Tons bzw. sein Notenwert bestimmt. Das NoteOff-Event wird nun nicht länger benötigt. Abschließend wird die Note des MidiEvents, nun mit entsprechendem Notenwert versehen, in die MusicXML-Datei eingetragen. Handelt es sich bei der Note um eine Akkordnote, so wird mit dem NoteOff-Event der gesamte Akkord geschlossen und alle Töne des Akkords in der MusicXML-Datei hinzugefügt. Die Dauer des Akkords wird also durch den Ton bestimmt, dessen Taste auf der Klaviatur als erste losgelassen wird. Alle Töne im Akkord bekommen unabhängig von ihrer eigentlichen Dauer den gleichen Notenwert zugeordnet.

Bestimmung der Tondauer bzw. des Notenwerts Der Notenwert des Tons wird in Abhängigkeit von der festgelegten Dauer einer ganzen Note definiert. Die Dauer einer ganzen Note lässt sich aus dem vom Nutzer eingestellten Tempo (Anzahl der Viertelnoten pro Minute) herleiten. Zur Bestimmung des Notenwerts wird die Dauer der ganzen Note als Richtwert abhängig von der Anzahl der insgesamt zu verwendenden Notenwerte (hier: Ganze, Halbe, Viertel und Achtel) in entsprechend viele Intervalle eingeteilt. Diese Intervalle definieren die Toleranzbereiche, in welche die Tondauer (Zeit, die zwischen dem NoteOn- und dem NoteOff-Event eines Tons vergangen ist) eingeordnet wird, um den entsprechenden Notenwert zuzuweisen. Die Intervallgröße verringert sich dabei mit abnehmendem Notenwert. Das Intervall, welches auf ganze Noten abgebildet wird, ist größer als das Intervall für Achtelnoten. Die Methode für die Bestimmung des Notenwerts setzt außerdem die passenden Werte für die (MusicXML) „duration“ und den „type“ der Note.

Verwaltung von Takten Die Unterscheidung und somit Verwaltung von verschiedenen Takten erfolgt nur im Tonfolgemodus. Im Akkordmodus

werden die Noten alle in einen MusicXML-Takt geschrieben und später vom Renderer zu Akkorden gruppiert.

Damit an der richtigen Stelle ein Takt beendet und ein neuer begonnen werden kann, muss zunächst der Wert bestimmt werden, bei dem ein Takt vollständig gefüllt ist. Dieser lässt sich aus der Taktart herleiten. Außerdem läuft für den jeweils aktiven Takt ein Zähler mit, welcher die Dauer der eingegebenen Töne aufsummiert. Dies geschieht am Ende der Methode zur Bestimmung der Notenwerte. Für Noten eines Akkords im Tonfolgemodus wird jeweils nur einmal die Tondauer zum Zähler addiert. Ist ein Takt voll, so wird in der MusicXML-Datei ein neuer, leerer Takt angelegt, in welchen von nun an die Noten eingetragen werden. Wird ein Ton eingegeben, dessen Tondauer zur „Überfüllung“ des aktuellen Taktes führt, so wird der Ton in zwei Noten gleicher Tonhöhe gespalten. Die erste Note bekommt als Dauer den Wert zugewiesen, welcher zum Auffüllen des aktuellen Taktes fehlt und wird diesem Takt hinzugefügt. Die zweite Note erhält als Dauer den Differenzwert zwischen der ursprünglichen Tondauer und der Dauer der ersten Note und wird in einen neuen, leeren Takt eingefügt. Die beiden Teilnoten werden dann durch einen Haltebogen verbunden. Beim Spalten der Noten kann es passieren, dass der Differenzwert einem Notenwert entspricht, der nicht durch eine einzige Note dargestellt werden kann (z.B. $\frac{1}{2} - \frac{3}{8} = \frac{5}{8}$; dies entspricht der Teilung einer halben Note in eine punktierte Achtel und eine Restwert). Für dieses Problem wurde noch keine adäquate Lösung gefunden. Auch können punktierte Noten aktuell nur durch die Aufspaltung einer Note am Taktende entstehen.

Eingaben korrigieren Neben den schon vorgestellten Methoden bietet der *Parser* zwei verschieden Funktionen um beispielsweise falsche Noteneingaben zu korrigieren. Die *backspace*-Funktion erlaubt das Löschen der zuletzt eingegebenen Note (aus der MusicXML-Datei und damit aus der Ausgabe). Falls noch keine Noten eingegeben wurden, geschieht nichts. War die gelöschte Note die Note am Ende eines Taktes, so wird der zuvor schon neu angelegte, leere Takt in der MusicXML-Datei ebenfalls wieder gelöscht.

Die *reset*-Funktion hingegen ermöglicht das Zurücksetzen der gesamten Eingabe, indem sie alle eingegebenen Noten wieder (aus der MusicXML-Datei) löscht. Dabei werden auch alle erzeugten Takte (abgesehen vom ersten Takt) wieder entfernt.

3.5 Die visuelle Ausgabe

Im Folgenden wird beschrieben wie die *Renderer*-Klasse aus den erzeugten MusicXML-Daten eine visuelle Repräsentation der Noten erstellt.

3.5.1 Der Renderer

Die Klasse *„Renderer“* ist dafür zuständig aus den vorliegenden MusicXML-Daten die optische Ausgabe der Anwendung zu generieren und darzustellen. Dies geschieht unter Verwendung der *FreeType2*-Softwarebibliothek. Mit dieser werden aus einer gegebenen Schrift-Datei im TTF-Format („MusiSync.ttf“ (siehe Abschnitt 3.5.1) die erforderlichen Texturen zur Darstellung der verschiedenen Notenzeichen generiert. Diese werden anschließend unter Verwendung eines passend positionierten Rechtecks, auf welches die Texturen abgebildet werden, gerendert. Die Positionierung des Rechtecks geschieht durch Angabe der Position der unteren linken Ecke im Fenster, die wie eine Art Cursor während des Renderns in x- und y-Richtung verschoben wird. Als Hilfsmittel kommen *OpenGL*[21], *GLFW*[22] für die Fensterverwaltung sowie die *stb_image-Bibliothek*[23] zum Laden der verwendeten Texturen zum Einsatz.

Der Konstruktor Der Konstruktor dieser Klasse setzt zunächst die Standardwerte der Membervariablen. Dazu zählen die Fenstergröße und -position, Skalierungswerte für die globale Anpassung der Größen der gerenderten Texturen oder auch die Schriftgröße, die bei der Generierung der Texturen durch FreeType2 genutzt wird. Anschließend werden die Geometrie der Notensysteme durch Festlegen der Endpunkte der jeweiligen Linien initiiert. Auch der *Renderer* wird in seinem Konstruktor über eine Referenz mit einem XMLManager-Objekt verbunden. Des Weiteren wird das spätere Ausgabefenster mit Hilfe von GLFW erzeugt, der OpenGL-Kontext gesetzt und ein Shader-Objekt (siehe Abschnitt 3.5.1) zum Verwalten der Shader erzeugt. Zum Schluss werden die Texturen für die Darstellung der Notenzeichen generiert (siehe Abschnitt 3.5.1).

Generieren der Texturen Die Texturen für die einzelnen Notenzeichen werden mit der *FreeType2*-Bibliothek aus der „MusiSync.ttf“ erzeugt. Dazu wird zuerst eine FT_Library angelegt. Anschließend wird für den Zugriff auf die Schrift-Datei eine FT_Face-Objekt erzeugt. Dann können die Daten aus der gegebenen Schrift-Datei in das FT_Face geladen werden. Als Nächstes wird für jeden in das FT_Face geladenen Character aus der Schrift eine Bitmaptextur generiert. Zusätzlich wird für jede dieser Texturen ein OpenGL-Texture-Handle angelegt und die OpenGL-Texturparameter gesetzt. Das Texture-Handle wird zusammen mit den weiteren Angaben zur Textur[24] in einer speziellen Datenstruktur gespeichert. Alle Instanzen die-

[21] Homepage: `https://www.opengl.org/`
[22] Homepage: `http://www.glfw.org/`
[23] GitHub: `https://github.com/nothings/stb`
[24] z.B. Angaben zur Ausdehnung der Textur in x- und y-Richtung, Koordinaten des oberen linken Eckpunktes

ser Datenstruktur werden für den späteren Zugriff in einer *std::map* abgelegt, welche den entsprechenden Character der Schrift auf die so gespeicherte Textur abbildet.

Renderfunktion Diese Renderfunktion wird in jedem Durchlauf der Renderschleife in der *main*-Methode aufgerufen. Als Erstes werden die zuvor gerenderten Daten aus dem Datenspeicher des *Renderers* gelöscht und die neuen Daten aus der MusicXML-Datei geladen. Im Anschluss wird die Geometrie der Notensysteme gerendert. Im Notenblattmodus werden vier Systeme dargestellt, im Durchlaufmodus nur eines. Bis zu diesem Punkt war die Verwendung von Texturen noch nicht nötig, nun wird diese jedoch im Shader durch Setzen der bool-Variablen aktiviert. Außerdem wird die „Cursor-Position“ in horizontaler und vertikaler Richtung auf den Zeilenanfang und die erste Linie des obersten Notensystems gesetzt. Das weitere Vorgehen der Renderfunktion unterscheidet sich je nach aktiviertem Darstellungsmodus.

Renderfunktion: Tonfolgemodus Im Tonfolgemodus werden zunächst die Attribute des ersten Taktes (einziger Takt, der Attribute speichert) als Notenzeichen (siehe Abschnitt 3.5.1) durch jeweiliges Versetzen des Cursors in y-Richtung in jedem Notensystem gerendert. Dabei wird der Cursor jeweils in x-Richtung weiter gesetzt. Anschließend werden mit Hilfe zweier geschachtelter Schleifen alle Noten aller Takte (siehe Abschnitt 3.5.1) dargestellt. Nach jedem Takt (abgesehen vom Zeilenende) wird ein Taktstrich gezogen, indem der Geometrie der Taktstriche weitere Punkte mit der aktuellen y-Position hinzugefügt werden. Abschließend werden die Geometrien der Hilfslinien und Taktstriche gerendert.

Renderfunktion: Akkordmodus Im Akkordmodus werden alle Noten in der MusicXML-Datei in einen einzigen Takt geschrieben. Zunächst werden die Attribute dieses Taktes in jedem Notensystem gerendert. Hierbei wird die Taktangabe ignoriert, da im Akkordmodus keine Takte dargestellt werden. Dann werden durch zwei geschachtelte Schleifen jeweils eine vom Nutzer über die grafische Benutzeroberfläche angegebene Anzahl an Noten zu einem Akkord zusammengefasst. Innerhalb eines Akkords wird der Cursor nur in y-Richtung verschoben, damit die Akkordnoten übereinander dargestellt werden. Ist ein Akkord abgeschlossen, wird der Cursor in x-Richtung weiter gerückt. Anschließend wird die Geometrie der gegebenenfalls nötigen Hilfslinien gerendert.

Das Rendern der einzelnen Noten im Akkordmodus funktioniert genauso wie das Rendern der Noten im Tonfolgemodus (siehe Abschnitt-3.5.1). Der einzige Unterschied besteht darin, dass im Akkordmodus nicht

die berechneten Notenwerte, sondern alle Töne ausschließlich als ganze Noten angezeigt werden.

Renderfunktion: Einzelne Note Falls es sich bei der aktuell darzustellenden Note um eine Akkordnote im Tonfolgemodus handelt, wird der Cursor in x-Richtung, der zuvor wie immer weiter gerückt wurde, zurückgesetzt, damit die Noten des Akkords übereinander dargestellt werden. Danach wird der Versatz in y-Richtung der Tonhöhe der Note entsprechend aus der Datenbank (siehe Abschnitt 3.3.2) ausgelesen und der Cursor in der Vertikalen demgemäß verschoben. Falls die Note ein Vorzeichen besitzt wird dieses nun als Notenzeichen (siehe Abschnitt 3.5.1) gerendert. Es werden automatisch nur Kreuze angezeigt, da die enharmonische Verwechslung nur anhand der MIDI-Daten als Eingabe nicht ohne Weiteres aufgelöst werden kann. Dazu bedarf es weiterer Nutzereingaben, die aktuell noch nicht in der Anwendung implementiert sind. Der nächste Schritt ist die unter Umständen nötige Anpassung der Geometrie der Hilfslinien. Ist der Versatz der Note in vertikaler Richtung größer als vier oder kleiner als minus eins (in Relation zur Basis[25] des Notensystems), so werden entsprechend viele Hilfslinien durch Hinzufügen von Punkten zur Hilfsliniengeometrie eingefügt. Es sind bis zu drei Hilfslinien jeweils nach oben und unten möglich. Dann wird aus der Datenbank (siehe Abschnitt 3.3.2) derjenige Character ausgelesen, der dem Notentyp der Note entspricht. Dieser wird für den Zugriff auf die passende Textur benötigt. Außerdem wird ebenfalls aus der Datenbank (siehe Abschnitt 3.3.2) die schrift- und Zeichen-spezifische Skalierung ermittelt. Nun kann das Notenzeichen als „Text“ durch die passende Textur und mit passender Skalierung gerendert werden (siehe Abschnitt 3.5.1)). Dabei wird die Darstellung der Note um 180 Grad gedreht, falls ihre vertikale Position überhalb der dritten Notenlinie im Notensystem liegt. Handelt es sich bei der abzubildenden Note um eine Note mit Fähnchen (Achtel oder kleinerer Notenwert), so muss stattdessen eine entsprechend angepasste Textur, welche die korrekt gedrehte Note mit den Fähnchen auf der rechten Seite des Notenhalses beinhaltet, mit Hilfe des passenden Characters gewählt werden. Falls es sich um eine gebundene Note am Taktende handelt wird nun noch der Haltebogen (ebenfalls in Form eines Notenzeichens) abgebildet. Der Cursor wird in horizontaler Richtung weiter gesetzt. Zum Schluss wird gegebenenfalls ein Zeilenumbruch (siehe Abschnitt 3.5.1) eingefügt.

Rendern eines Notenzeichens Das Rendern eines Notenzeichens funktioniert wie das Rendern eines „normalen“ Textes mit der FreeType2-Bibliothek. Der „Text“ besteht in diesem Fall allerdings nur aus einem einzigen

[25] Als Basis wird der Zwischenraum zwischen der ersten und zweiten Notenlinie des Systems angenommen

Character. Dieser wird der Funktion als Parameter übergeben. Mit Hilfe des Characters werden die Texturdaten aus der *std::map*, in die sie bei ihrer Generierung abgelegt wurden, ausgelesen. Im nächsten Schritt werden die Eckpunkte des Rechtecks, auf das die Textur abgebildet werden soll, gemäß der Texturgröße und der aktuellen Position des Cursors im Fenster gesetzt. Die Textur wird gebunden und im Anschluss wird das Rechteck gerendert. Rückgabewert der Funktion ist der horizontale Versatz des Cursors auf die Position, an der das nächste Zeichen gerendert werden darf.

Zeilenumbrüche Die Methode für die Umsetzung der Zeilenumbrüche unterscheidet zwischen dem Notenblattmodus und dem Durchlaufmodus. Im Notenblattmodus wird, wenn das Ende einer Notenzeile erreicht worden ist, der Cursor in y-Richtung um ein Notensystem nach unten versetzt. Im Durchlaufmodus werden in diesem Fall alle dargestellten Noten um eine Stelle nach links gerückt. Die Noten am Anfang der Zeile verschwinden aus der Anzeige und am Zeilenende wird so Platz für neue Noten frei. Dies wird über zwei Indizes gelöst, die angeben ab welcher Note in welchem Takt die Töne gerendert werden sollen.

Aktualisierung der Darstellungsmodi Werden über die grafische Benutzeroberfläche andere Darstellungsmodi eingestellt, so können diese über eine spezielle Methode an den *Renderer* weitergegeben werden. Dieser setzt zum einen die Hilfslinien und Taktstriche zurück, das heißt sie werden gelöscht und entsprechend für den neuen Darstellungsmodus gesetzt (z.B. Akkordmodus ganz ohne Taktstriche). Außerdem wird die Geometrie der Notensysteme aktualisiert, sodass für den Tonfolgemodus stets vier Systeme und für den Durchlaufmodus nur ein System angezeigt wird.

Löschen von Noten Auch der *Renderer* stellt äquivalent zum Parser eine Methode für das Löschen der letzten eingegebenen Note bereit. Diese löscht die letzte Note aus der Liste der darzustellenden Noten. Dazu gehört auch, dass die Geometrie der Hilfslinien angepasst wird, falls diese Note Hilfslinien besaß, bzw. dass ein Taktstrich gelöscht wird, falls die Note die letzte in einem Takt war.

Schriftart Bei der verwendeten Schriftart handelt es sich um die Schrift „MusiSync“ von der Plattform Fontspace. Die Schrift liegt im TTF-Format vor. Sie beinhaltet alle grundlegenden Zeichen für Noten- und Pausenwerte sowie Notenschlüssel, viele verschiedene Taktarten, Versetzungszeichen etc. Größtes Problem bei dieser Schrift ist das Fehlen von Notenzeichen zur korrekten Notation von Sechszehnteln und kleineren Notenwerten, falls diese mit dem Hals nach unten gezeichnet werden sollen oder im gleichen Zusammenhang die korrekte Notation von Punktierungen. Es wird

deshalb für den späteren Praxiseinsatz in Erwägung gezogen, eine andere Schriftart einzubinden. Die Schrift wurde gewählt, da sie eine der wenigen frei verfügbaren Schriftarten ohne Notenlinien im Hintergrund der Zeichen ist. „MusiSync“ ist unter der SIL Open Font Lizenz (OFL) veröffentlicht und somit auch für die kommerzielle Nutzung freigegeben.

Hilfslinien und Taktstriche Die Hilfslinien und Taktstriche sind, wie das Notensystem, von „Geometry“ abgeleitete Klassen. Geometry ist eine Oberklasse, die die Vertex-Arrays und Vertex-Buffer, sowie eine Liste der zugehörigen Punkte etc. verwaltet. Während des Rendervorgangs werden den Vertex-Listen der Hilfslinien und Taktstrichen jeweils entsprechend neue Punkte hinzugefügt. Die Hilfslinen und Taktstriche werden als *GL_LINES* gerendert.

Die Shader Die Shader-Klasse wurde aus dem CVK-Framework der AG Computergrafik der Universität Koblenz-Landau entnommen und übernimmt das Laden des Quellcodes der Shader sowie das Generieren und Binden der Shader an ein Programm. Die Shader an sich sind sehr minimalistisch gehalten. Für die Anwendung wird nur ein Vertex- und ein Fragment-Shader benötigt. Der Vertex-Shader gibt dabei nur die Position und die Texturkoordinaten des Vertex an den Fragment-Shader weiter. Der Fragment-Shader setzt die Fragmentfarbe dann entweder auf einen von der Anwendungsseite übergebenen Farbwert oder liest den Wert an der entsprechenden Stelle aus einer Textur aus. Ob die Textur verwendet werden soll, oder nicht kann durch einen booleschen Wert (*bool useTexture*) gesteuert werden.

3.6 Die Verknüpfung der Komponenten

Bisher wurden zwei theoretisch voneinander unabhängige Komponenten des Systems entwickelt. Im nächsten Abschnitt soll nun beschrieben werden wie diese beiden Komponenten durch eine Verwaltungseinheit für die MusicXML-Dateien zusammengefügt wurden.

3.6.1 Der XML-Manager

Die *XMLManager*-Klasse bildet das Bindeglied zwischen der Eingabe der MIDI-Daten in Reader und Parser und der visuellen Ausgabe durch den Renderer. Er verwaltet die MusicXML-Datei. In diese werden die musikalischen Daten einerseits vom Parser eingetragen. Andererseits wird sie vom Renderer zur Erfassung der darzustellenden Daten geladen. Für den Zugriff (das Lesen und Schreiben) auf diese Datei verwendet der *XML-Manager* die TinyXML-2-Bibliothek (siehe Abschnitt 2.3.2). Dazu besitzt die

Klasse ein *XMLDocument*-Objekt. Der Parser und der Renderer verwenden jeweils ein und dieselbe *XMLManager*-Instanz.

Der Konstruktor dieser Klasse setzt zunächst den Pfad unter dem die MusicXML-Datei zu erreichen sein soll und erzeugt dann ein leeres *XML-Document*. Dann wird eine Standard-Datei (ähnlich dem Beispielcode in 2.2.2) mit einem MusicXML-konformen Header, dem Wurzelelement der Datei, der „part-list“ und dem einzigen „part“ sowie einem leeren Takt („measure“) mit Standardattributen (für Notenschlüssel, Tonart, Takt etc.) geschrieben. Aktuell ist der erste Takt in der Datei auch der einzige Takt mit Attributen.

Neben dem Konstruktor bietet die *XMLManager*-Klasse Methoden für das Laden und Abspeichern der MusicXML-Datei.

Außerdem wird eine Funktion zum Laden der MusicXML-Daten in eine vom Renderer verarbeitbare Datenstruktur (siehe Abschnitt 3.3.1) bereitgestellt. Diese Methode lädt zuerst das *XMLDocument* und schreibt die Daten der Datei taktweise und notenweise in die Datenstruktur des Renderers.

Alle anderen Methoden der Klasse sind nach dem selben Prinzip aufgebaut. Sie laden als Erstes das *XMLDocument*, führen dann die entsprechenden Befehle auf der Datei aus und speichern das Dokument schließlich wieder unter dem gesetzten Dateipfad ab.

Diese Methoden bieten folgende Funktionalitäten:

- Veränderung der Taktattribute
- Einfügen einer normalen Note
- Einfügen einer Note mit Haltebogen
- Einfügen eines neuen, leeren Taktes
- Löschen des zuletzt eingefügten (leeren) Taktes
- Löschen der zuletzt eingefügten Note
- Löschen aller Noten in der Datei
- Kopieren der Taktattribute in den nachfolgenden Takt

Sämtliche Funktionen der Klasse sind so aufgebaut, dass sie die ihnen entsprechende Methode aus *XMLTools* (siehe Abschnitt 3.6.1) aufrufen, welche dann die jeweiligen Operationen mit Hilfe der *TinyXML-2*-Bibliothek ausführen.

Die Methoden des *XMLManagers* werden jeweils im Parser und im Renderer aufgerufen.

XMLTools *XMLTools* ist eine hpp-Datei, welche die eigentliche Funktionalität für den Zugriff und die Bearbeitung der MusicXML-Datei bereitstellt. Für fast jede Methode des *XMLManagers* gibt es eine entsprechenden Methode in *XMLTools*. Diese Funktionen parsen das Dokument via *TinyXML-2* und navigieren jeweils durch die XML-Hierarchie zur gewünschten Position im Dokument. Dort werden die gefragten Daten ausgelesen, überflüssige Teile der Datei gelöscht oder neue oder veränderte Daten geschrieben.

3.7 Die grafische Benutzerschnittstelle

Die *GUI*-Klasse ist für die Umsetzung einer grafischen Benutzeroberfläche für die Steuerung der Applikation zuständig. Sie verwendet die *ImGui*-Bibliothek und wurde zu großen Teilen aus dem Projekt „GeKo - GameEngineKobenz“[26] übernommen. Die *GUI*-Klasse besitzt eine Referenz auf ein eigenes *GLFWwindow*, in welches die Benutzeroberfläche gerendert wird. Außerdem speichert jedes *GUI*-Objekt eine Liste (in Form eines *std::vector*) der einzelnen zu ihm gehörenden *GUI*-Elemente. Diese Elemente werden in der *main*-Datei angelegt und zu einer *GUI*-Instanz hinzugefügt.

Der Konstruktor setzt Default-Werte für die Member-Variablen und initiiert das *GLFWwindow*.

Die *Render*-Methode läuft mit einer Schleife über alle *GUI*-Elemente und rendert diese im Anzeigefenster.

Des Weiteren stellt die Klasse Funktionen zum Hinzufügen, Ausblenden, Anzeigen und zum Abfragen der Sichtbarkeit von *GUI*-Elementen bereit.

Die zur Verfügung stehenden *GUI*-Elemente werden in einer eigenen hpp-Datei definiert. Diese liefert für jede verwendbare *GUI*-Komponente eine eigene Klasse, welche von der Oberklasse „Element“ abgeleitet wird. Durch Hinzufügen weiterer Klassen in dieser Datei kann die Funktionalität der grafischen Benutzeroberfläche erweitert werden. Derzeit fasst die hpp-Datei die folgenden *GUI*-Komponenten zusammen:

- Text: eine einfache Textausgabe in der Benutzeroberfläche
- SliderFloat und SliderInt: ein Schieberegler für die Veränderung einer Float- oder Integer-Variablen in einem gegebenen Intervall
- Header: eine Komponente zum Gruppieren beliebiger Elemente
- Spacing und Separator: Komponenten zum Einfügen von Abständen und Trennlinien zwischen *GUI*-Elementen

[26] *GeKo* wurde im Rahmen eines Projektpraktikums der AG Computergrafik an der Universität Koblenz-Landau im Wintersemester 2014/2015 entwickelt. Der Code steht zur freien Verwendung auf GitHub zur Verfügung

- PushButton: eine Komponente zur Realisierung eines drucksensiblen Knopfes
- RadioButton und CheckBox: Komponenten für Optionsfelder (RadioButton: Auswahl von genau einer Option, CheckBox: Mehrfachauswahl möglich)
- PickList: eine Komponente für der Realisierung einer Auswahlliste (Auswahl von genau einem Eintrag aus einer gegebenen Liste)

3.8 Das Zusammenbringen der Komponenten in der *main*-Datei

In der *main*-Datei werden alle einzelnen Komponenten der Anwendung zusammengefügt. Sie beinhaltet die *main*-Methode inklusive der eigentlichen Renderschleife.

Die *main*-Datei kennt alle Elemente: den Reader, den Parser, den Renderer, den XMLManager und die GUI. Außerdem sind hier feste Werte für die Fensterhöhe und Fensterbreite sowohl des GUI-Fensters, als auch des Hauptfensters der Anwendung definiert.

Die *main*-Methode ist wie folgt aufgebaut:

Zunächst wird eine Reader-Instanz erzeugt. Dann wird die GUI initialisiert.

Darauf folgt eine erste Renderschleife, welche dafür sorgt, dass die grafische Benutzeroberfläche im Einstellungsfenster angezeigt wird, sodass der Nutzer die initialen Einstellungen setzen kann. Dazu wird für jeden Durchlauf überprüft, welche GUI-Elemente gerade angezeigt werden müssen, bevor diese dann gerendert werden. Die Schleife wird abgebrochen, sobald die Eingaben durch Betätigen des „OK“-Knopfes bestätigt wurden, oder das Fenster geschlossen wurde.

Wurde das GUI-Fenster geschlossen worden, werden das GUI- und das Reader-Objekt gelöscht und die Anwendung beendet. Ansonsten werden nun die eingegebenen Daten aus der GUI ausgelesen und anschließend über die zuvor erzeugte Reader-Instanz der gewünschte MIDI-Port geöffnet. Als Nächstes wird ein XMLManager-, ein Parser- und ein Renderer-Objekt angelegt und die jeweils benötigten GUI-Daten an den Parser und den Renderer übergeben.

Darauf folgt die Hauptrenderschleife. Diese stellt zunächst wieder die GUI dar und fragt im Anschluss ab, ob einer der Knöpfe der GUI gedrückt worden ist. Wurde der „OK“-Knopf gedrückt, so werden die GUI-Daten neu ausgelesen und gegebenenfalls der Parser und der Renderer aktualisiert sowie unter Umständen ein anderer MIDI-Port geöffnet. Ein Drücken des „RESET“-Knopfes bewirkt das Zurücksetzen des Parser- und des Renderer-Objektes, das Betätigen des „ZURUECK“-Knopfes das Löschen der

letzten Eingabe sowohl im Parser als auch im Renderer. Dann werden eventuell eingegebene MIDI-Daten über den Reader eingelesen und mitsamt ihres zugehörigen Zeitstempels an den Parser übergeben. Falls MIDI-Daten erfasst wurden, werden diese jetzt vom Parser übersetzt und in MusicXML-Daten umgewandelt. Abschließend werden die Daten aus der MusicXML-Datei vom Renderer im Hauptfenster der Anwendung angezeigt.

Die Hauptrenderschleife wird abgebrochen, sobald eines der beiden Fenster (GUI-Fenster oder Hauptfenster) geschlossen wurde.

Schließlich wird auch das jeweils andere Fenster geschlossen und alle erzeugten Instanzen gelöscht.

4 Evaluation

Um zu überprüfen, inwiefern die Anforderungen an das System bereits erfüllt sind, wurde eine erste Version der Anwendung einer Evaluation durch den zukünftigen Anwender, Herrn Lutz Brenner, unterzogen. Die Evaluation wurde in Form einer Anwenderbefragung anhand eines eigens entwickelten Fragebogens (siehe Anhang) durchgeführt. Dieser Fragebogen und die Ergebnisse der Bewertung werden hiernach erläutert.

4.1 Ablauf der Evaluation

Zu Beginn der Anwenderbefragung wurden die Funktionen des Systems und deren Handhabung während einer kurzen Demonstration veranschaulicht. Danach stand dem Probanden Zeit zum eigenständigen Testen der Applikation zur Verfügung. Anschließend sollte der Fragebogen ausgefüllt werden. Dieser gliedert sich in insgesamt acht Themenbereiche. Die Bewertung der einzelnen Aspekte erfolgt anhand einer Skala von 1 bis 5 (1 entsprach „sehr schlecht“, 5 entsprach „sehr gut“). Zusätzlich steht am Ende jedes Fragenblocks ein Kommentarfeld zur Verfügung.

Vorweg wird die Testperson aufgefordert ihre *Erwartungen* an das Programm zu formulieren.

Der nächste Abschnitt thematisiert die *MIDI-Eingabe*. Es wird gefragt, als wie intuitiv die Bedienung der Software mit dem MIDI-Keyboard empfunden wird.

Weiter soll die *Optik* der Notenschrift und der Notenlinien bewertet werden.

Der nachfolgende Fragenblock bezieht sich auf die verschiedenen *Darstellungsmodi*. Es soll begutachtet werden, wie die einzelnen Darstellungsmodi gefallen, wie flüssig das Wechseln der einzelnen Modi funktioniert und welcher Modus am hilfreichsten war.

Der Proband wird dann aufgefordert, die *Korrektheit* der Darstellung von einzelnen Elementen einzustufen.

Danach schließt sich der Themenbereich *„Performanz“* an. Die Leistungsfähigkeit einzelner Teile (Eingabe der MIDI-Daten, Bildschirmausgabe) sowie der Anwendung insgesamt soll bewertet werden.

Nun soll die *Benutzeroberfläche* betrachtet werden. Zur Evaluation stehen Punkte wie die Optik, die Übersichtlichkeit und der Gesamteindruck der GUI. Abschließend wird ein *Gesamturteil* über das System erbeten. Neben dem Nutzen der Anwendung bei der Unterrichtsgestaltung, soll die Gesamtzufriedenheit und die Erfüllung der Erwartungen beurteilt werden. Außerdem soll eine Einschätzung zur Wichtigkeit der möglichen Erweiterungen zum Programm gegeben werden. Zum Schluss stehen noch einmal zwei Kommentarfelder für Anmerkungen zu besonders guten bzw. besonders schlechten Aspekten zur Verfügung.

4.2 Ergebnisse der Evaluation

Insgesamt ergab die Auswertung der Evaluation ein äußerst positives Resultat. In nahezu allen Bereichen wurden beste Bewertungen von 4 bis 5 auf der Skala erzielt. Einzig ein Unterpunkt im Teilgebiet „Genauigkeit/-Korrektheit“ erreicht nur eine mittelmäßige Benotung.

Lutz Brenner erwartete von der Applikation eine einfache Darstellung von Noten, Rhythmen und Akkorden sowie die Möglichkeit zur Abspeicherung von eingegebenen Notendiktaten.

Die Eingabe der Noten über das MIDI-Keyboard wurde als sehr intuitiv empfunden. Besonderes Lob erhielt die „sehr einfache Bedienung“, die „schöne Oberfläche“ und die „selbsterklärende Darstellung“.

Die Optik der Notenschrift und Notenlinien gefiel Herrn Brenner ebenfalls sehr gut.

Gleichermaßen bewertete er die Darstellungsmodi allesamt mit Bestnoten. Das Wechseln der Modi funktionierte flüssig. Als am Hilfreichsten wurden der Tonfolgemodus in Verbindung mit dem Notenblattmodus erachtet.

Die Darstellung der Tonhöhen, Tonlängen, Taktstriche, Hilfslinien und Akkorde war korrekt. Mängel wies einzig die Darstellung der Versetzungszeichen, welche sich auf die Wiedergabe von Kreuzen beschränkt, auf. Herr Brenner schlug vor eine Möglichkeit für die Vorauswahl der Versetzungszeichen b, Doppel-b sowie Kreuz und Doppel-Kreuz über die grafische Benutzeroberfläche anzubieten. Dabei sollte es möglich sein, erst ein Zeichen auszuwählen und anschließend eine Note (gegebenenfalls auch nur über die weißen Tasten der Klaviatur) einzugeben, die dann mit dem gewünschten Versetzungszeichen angezeigt wird. Außerdem merkte Lutz Brenner an, dass es sinnvoll wäre genauso eine Vorauswahl von festen Notenwerten in der Benutzeroberfläche zu integrieren, um die Darstellung der korrekten bzw. gewünschten Notenwerte zu erleichtern.

Die Anwendung lief während des Testdurchlaufs sehr flüssig und alle eingegebenen Noten wurden auch auf der Anzeige wiedergegeben. Die Bildschirmausgabe reagierte in den meisten Fällen angemessen schnell auf die Eingaben des Keyboards.

Lutz Brenner empfand auch die grafische Benutzeroberfläche als gut gelungen. In den Bereichen Optik, Übersichtlichkeit und intuitive Bedienung erreichte auch sie jeweils die volle Punktzahl. Einziger Kritikpunkt an der Handhabung des Einstellungsmenüs war, dass alle Eingaben durch Betätigen der „OK"-Taste erst autorisiert werden müssen, bevor sie umgesetzt werden. Es wurde die direkte Darstellung der geänderten Einstellungen ohne zusätzliche Bestätigung gewünscht.

Die Angaben des Probanden im Abschnitt „Gesamteindruck der Anwendung" bekräftigten den allgemeinen Tenor der bisherigen Bewertung. Herr Brenner ist überzeugt, dass ihm die Anwendung bei der Unterrichtsgestaltung von großem Nutzen sein wird. Er war mit dem System im Großen und Ganzen recht zufrieden und seine Erwartungen wurden größtenteils erfüllt (jeweils 4-5 von 5 Punkten). Besonders wichtig sind ihm das Hinzufügen von Pausendarstellungen, einer Einstellungsmöglichkeit für feste Notenwerte sowie einer Funktion zum Speichern und Laden von eingespielten Daten zur Anwendung. Als eher zweitrangig erachtet der Proband die Integration einer akustischen Ausgabe der Noten, einer Einstellungsmöglichkeit für verschiedene Tonarten oder die Notendarstellung im Klaviersystem.

Spezielle Anerkennung fanden im Kommentarfeld am Ende neben dem Akkordmodus und dem „übersichtlichen Bedienfeld„, die „gute Darstellung" sowie die „einfache Bedienung". An dieser Stelle wurde auch noch einmal insbesondere das Fehlen von einer Speicherfunktion und einer Darstellungsmöglichkeit von Pausen betont.

In einem an die eigentliche Evaluation anknüpfenden Gespräch mit Herrn Brenner erläuterte dieser einige zusätzliche Funktionalitäten, um die das System erweitern werden könnte. Es wäre sinnvoll, wenn über die grafische Benutzeroberfläche neben Pausen und Versetzungszeichen auch Vorzeichen am Beginn einer Notenzeile und Punktierungen einzelner Noten zur Darstellung hinzugefügt werden können. Außerdem wäre eine Art „Tonleitermodus" denkbar. Die Idee hinter diesem Modus lehnt stark an die derzeitig von Herrn Brenner für Unterrichtszwecke verwendete *Lichtorgel* (siehe Abschnitt 1.2) an. In der Ausgabe sollen ausgegraut alle im aktuell gesetzten Notenschlüssel darstellbaren Töne beispielsweise in Form von ganzen Noten angezeigt werden. Wird nun einer der Töne auf der Klaviatur des MIDI-Keyboards gedrückt, so soll die entsprechende Note in der Anzeige zum Beispiel durch Einfärben hervorgehoben werden. Wird die Taste wieder losgelassen, so wird die Hervorhebung des Tons wieder entfernt. Hierbei sollte auch das gleichzeitige Betätigen mehrerer Tasten möglich sein.

Insgesamt lässt sich zusammenfassen, dass die Anwendung die Ansprüche des zukünftigen Nutzers schon weitgehend erfüllt, er sich jedoch hauptsächlich mehr Möglichkeiten wünscht die Darstellung der Noten durch Eingaben über die Benutzeroberfläche anzupassen. Beispielsweise sollen Notenwerte, Vorzeichen und Versetzungszeichen, Punktierungen oder Pausen von außen ohne Verwendung des MIDI-Keyboards beeinflusst werden können.

5 Fazit und Ausblick

In dieser Arbeit wurde eine Software für die Darstellung von Noten im musikalischen Sinne entwickelt und anschließend im Rahmen einer Anwenderbefragung evaluiert.

Die Anwendung erhält die im MIDI-Format codierten Noten durch Einspielen dieser auf einen MIDI-fähigen Keyboard, welches via USB an das System angeschlossen wird, als Eingabe. Die MIDI-Daten werden dann in das MusicXML-Format überführt und zwischengespeichert. Ein eigens entworfener Renderer liest die abgespeicherte MusicXML-Datei wieder ein und setzt die Daten in eine grafische Wiedergabe der Töne in einem Notensystem inklusive Notenschlüssel und Taktangabe auf dem Bildschirm um.

Es wurden mehrere verschiedene Darstellungsmodi implementiert. Die Noten können zum einen als Tonfolge angezeigt werden. Dabei wird die Tondauer einer Note und damit ihr Notenwert automatisch bestimmt sowie eingespielte Akkorde durch die Anwendung erkannt und wiedergegeben. Zum anderen ermöglicht der Akkordmodus das schrittweise Aufbauen einzelner Akkorde. Die nacheinander eingespielten Noten werden dabei zu Akkorden mit jeweils einer bestimmten Anzahl an Tönen zusammengefasst. Es werden nur ganze Noten abgebildet.

Unabhängig davon, ob der Tonfolgemodus oder der Akkordmodus aktiv ist, können die Noten wahlweise in Form eines Notenblatts mit vier Notensystemen oder als eine durchlaufende Notenzeile dargestellt werden.

Die Eingabe und Wiedergabe der Töne im Tonfolge- und Akkordmodus wurden somit gemäß der Anforderungen verwirklicht. Mit dem „Durchlaufmodus“ wurde eine zusätzliche, nicht in den Anforderungen definierte Anzeigeform umgesetzt.

Die grafische Benutzeroberfläche der Applikation bietet dem Nutzer die Möglichkeit zusätzlich Informationen, die nicht im MIDI-Format enthalten sind, einzugeben. Dazu zählen der Notenschlüssel, die Taktart und das Tempo. Die Nutzerschnittstelle wurde wie gefordert schlicht gehalten und nur mit den nötigsten Einstellungsmöglichkeiten versehen.

Insgesamt lässt sich also sagen, dass die Implementierung des entwickelten Konzeptes die zu Beginn definierten Ansprüche an die Software

bis auf vereinzelte Aspekte erfüllt und an einigen Stellen, zum Beispiel mit dem Durchlaufmodus, sogar übersteigt.

Aus zeitlichen Gründen wurde auf die Darstellung von Pausen und das Einbinden einer akustischen Ausgabe der Noten verzichtet. Auch wurde die Möglichkeit, eingespielte Noten zu Speichern und zu einem späteren Zeitpunkt wieder zu laden nicht umgesetzt. Außerdem ist beispielsweise die für die (korrekte) Notation von kleineren Notenwerten (Achteln, Sechzehnteln etc.) erforderliche Balkennotation derzeit aufgrund der seriellen Abarbeitung der einzelnen Noten gleichermaßen noch nicht möglich.

Zur Evaluation dieser ersten Version der Software wurde eine Anwenderbefragung mit dem Experten und zukünftigen Anwender, Herrn Lutz Brenner, durchgeführt (siehe Abschnitt 4.2). Diese bestätigte im Rahmen dessen gleichermaßen, dass die Anwendung bereits die meisten seiner Anforderungen erfüllt.

Es wurde jedoch auch festgestellt, dass für den Einsatz in der Ausbildung der Nachwuchssängerinnen und -sängern noch weitere Komponenten eingefügt werden müssen.

Dem Programm fehlt es beispielsweise noch an Optionen, die Darstellung der Noten in Bezug auf Tondauern, Punktierungen, Versetzungszeichen etc. von außen ohne Verwendung des MIDI-Keyboards, sondern eher durch Eingaben über die grafische Benutzeroberfläche, anzupassen.

Des Weiteren zeigte das abschließende Gespräch mit dem Experten zusätzliche Erweiterungsmöglichkeiten für die Software auf. Diese wurden bereits in Kapitel 4.2 vorgestellt. Es wäre denkbar einen „Lichtorgelmodus“ umzusetzen, welcher das aktuell verwendete System des St. Martins-Chores (siehe Abschnitt 1.2) eins-zu-eins in eine digitale Version überführt.

Darüber hinaus wäre es vorstellbar auf Basis dieses Programms, insbesondere auf Basis des implementierten Renderers, einen vollständigen Editor für die eingespielten Noten zu entwickeln.

Literatur

[fon] *Fontspace*. http://www.fontspace.com/robert-allgeyer/musisync. – Zuletzt eingesehen im Dezember 2015

[fre] *The FreeType Project*. http://www.freetype.org/index.html. – Zuletzt eingesehen im Dezember 2015

[imG] *dear imgui*. https://github.com/ocornut/imgui. – Zuletzt eingesehen im März 2016

[imm] *Retained Mode Versus Immediate Mode*. https://msdn.microsoft.com/en-us/library/windows/desktop/ff684178(v=vs.85).aspx. – Zuletzt eingesehen im März 2016

[mida] *MIDI Association*. https://www.midi.org/. – Zuletzt eingesehen im Oktober 2015

[midb] *Summary of Midi Messages*. https://www.midi.org/specifications/item/table-1-summary-of-midi-message. – Zuletzt eingesehen im Februar 2016

[mXM] *MusicXML Tutorials*. http://www.musicxml.com/de/tutorial/. – Zuletzt eingesehen im Oktober 2015

[rtM] *The RtMidi Tutorial*. https://www.music.mcgill.ca/~gary/rtmidi/. – Zuletzt eingesehen im Dezember 2015

[stl] *Introduction to the Standard Template Library*. http://www.sgi.com/tech/stl/stl_introduction.html. – Zuletzt eingesehen im Februar 2016

[tin] *TinyXML-2 Documentation*. http://grinninglizard.com/tinyxml2docs/index.html. – Zuletzt eingesehen im Januar 2016

[Zie88] ZIEGENRÜCKER, Wieland: *Allgemeine Musiklehre*. 13. Auflage. Mainz : B. Schott's Söhne, 1988

Abbildungen

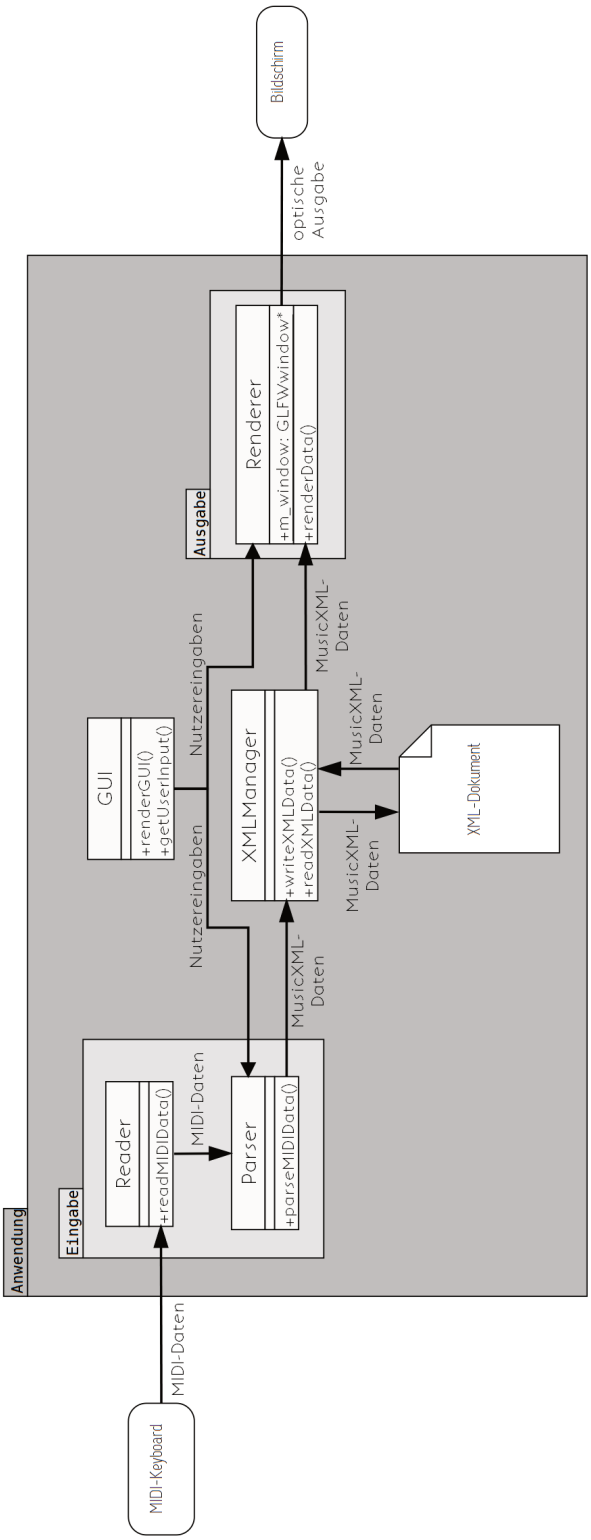


Abbildung 24: Verarbeitungspipeline der Anwendung

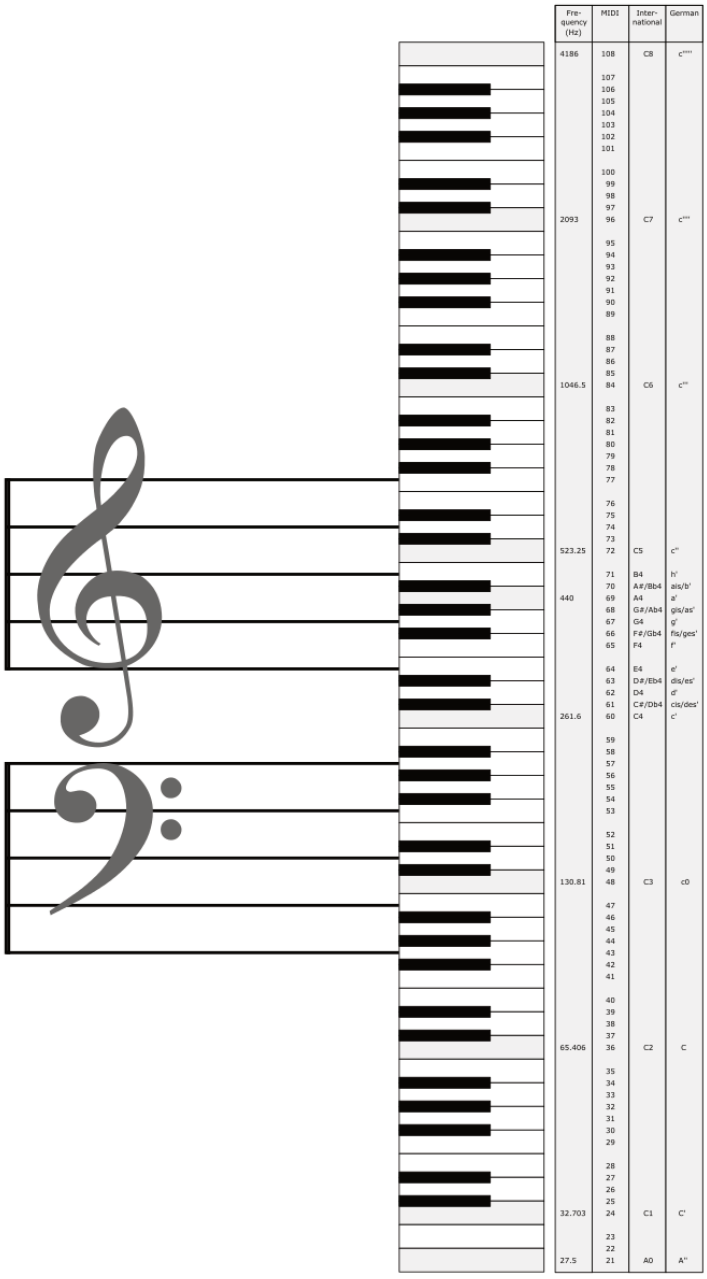


Abbildung 25: Keyboard mit zugehörigen *MIDI*-Nummern[27]

[27] Quelle: `https://de.wikipedia.org/wiki/Musical_Instrument_Digital_Interface`

Fragebogen zur Anwenderbefragung

Anleitung

UNIVERSITÄT KOBLENZ · LANDAU

Einführung

Die nachfolgende Anwenderbefragung dient zur Evaluation der im Rahmen meiner Bachelorarbeit „Eingabe und Darstellung von Noten für die musikalische Ausbildung von Kindern“ entwickelten Software zur Eingabe von Noten über ein MIDI-Keyboard und deren anschließende Darstellung in Notenschrift auf dem Bildschirm. Ihre Aufgabe als Experte und späterer Anwender ist es nun, besagte Software nach bestimmten Kriterien zu bewerten. Lesen Sie sich bitte hierzu zunächst die untenstehende Anleitung gründlich durch und befolgen sie, bevor Sie mit der Bewertung der Anwendung beginnen. Füllen Sie bitte anschließend den beiliegenden Fragebogen aus.

Anleitung

Bitte starten Sie zuerst das Keyboard und anschließend die Anwendung. Es erscheint am linken Bildschirmrand ein Fenster mit Einstellungen der Benutzeroberfläche. Diese Benutzeroberfläche dient einerseits dazu verschiedene Informationen einzugeben, die für die Darstellung der Noten wichtig sind, jedoch nicht aus den MIDI-Daten hergeleitet werden können. Andererseits können Sie hier zwischen unterschiedlichen Darstellungsmodi wechseln.
Im Folgenden werden die Ihnen zur Verfügung stehenden Einstellungen näher erläutert.

1. Wählen Sie das MIDI-Gerät aus, mit dem Sie eine Verbindung aufbauen wollen: Hier werden Ihnen die verfügbaren MIDI-Geräte angezeigt. Wenn mehrere Geräte angeschlossen sind, können Sie das gewünschte Gerät auswählen. Wenn keine Verbindung angezeigt wird, beenden Sie bitte die Anwendung, schließen Sie ein MIDI-Gerät an und starten Sie die Anwendung neu.

2. Wählen Sie den gewünschten **Notenschlüssel** aus: Sie können zwischen Violin- und Bassschlüssel wählen. (Standard: Violinschlüssel)

3. Wählen Sie die gewünschte **Taktart** aus: Sie haben die Wahl zwischen 16 verschiedenen Taktarten. (Standard: Viervierteltakt)

4. Wählen Sie das gewünschte Tempo aus: Mit dem Schieberegler können Sie einen ganzzahligen Wert zwischen 10 und 160 auswählen. Der Wert entspricht der Anzahl der Viertelnoten pro Minute. (Standard: 60)

5. Wählen Sie den gewünschten Darstellungsmodus aus: Hier haben Sie zum einen die Wahl zwischen dem Tonfolge- und dem Akkordmodus, zum anderen zwischen dem Notenblatt- und dem Durchlaufmodus. Aus beiden Gruppen muss je ein Modus selektiert werden. Die einzelnen Modi werden im Anschluss kurz erläutert. (Standard: Tonfolgemodus, Notenblattmodus)

Tonfolgemodus Im *Tonfolgemodus* werden die Noten als Tonfolge im Notensystem mit dem eingegebenen Notenschlüssel und der eingegebenen Taktart dargestellt. Die Töne werden angezeigt, sobald Sie die Taste auf der Klaviatur loslassen. Die Tonlänge wird automatisch dem Tempo und der Taktart angepasst. Außerdem werden im *Tonfolgemodus* einzelne Takte mit durch Taktstrichen unterteilt. Sobald ein Takt gemäß der eingestellten Taktart vollständig ist, wird ein Taktstrich gezogen. Das gleichzeitige Betätigen von mehreren Tasten des Keyboards führt auch im *Tonfolgemodus* zur Erzeugung eines Akkords. Die Länge der Töne in diesen Akkorden wird durch die Länge des Tons bestimmt, dessen zugehörige Taste auf der Klaviatur als erstes losgelassen wird. Der Akkord wird ebenfalls erst nach dem Loslassen dieser Taste angezeigt.

Akkordmodus Im *Akkordmodus* werden ausschließlich möglich Akkorde eingegeben. Die einzelnen Töne der Akkorde werden hierbei jedoch im Gegensatz zum Tonfolgemodus nacheinander eingegeben, d.h. es ist keine gleichzeitige Betätigung mehrerer Tasten der Klaviatur nötig. Entsprechend wird jeder Ton des Akkordes auch dargestellt, sobald seine Taste losgelassen wurde und nicht erst, wenn alle Töne

eingegeben sind. Hierzu werden Sie bei Auswahl des *Akkordmodus* zusätzlich dazu aufgefordert, die Anzahl der Noten, die zu einem Akkord zusammengefasst werden sollen, anzugeben. Es sind zwischen 1 und 5 Noten möglich. Die Länge der eingegebenen Töne wird in diesem Modus ignoriert und es werden ausschließlich ganze Noten angezeigt. Aus diesem Grund steht auch am Beginn der Notenzeilen auf dem Bildschirm im *Akkordmodus* nur ein Notenschlüssel und **keine** Taktart. Außerdem werden keine Taktstriche gezogen.

Notenblattmodus Im *Notenblattmodus* werden vier Notenzeilen im Anzeigefenster dargestellt. Wird das Ende einer Notenzeile erreicht, erzeugt die Anwendung automatisch einen Zeilenumbruch und fährt mit der Darstellung der Noten in der nächsten Notenzeile fort.

Durchlaufmodus Im *Durchlaufmodus* wird nur eine einzige Notenzeile im Anzeigefenster dargestellt. Wird das Ende der Notenzeile erreicht, werden die angezeigten Noten nach links gerückt, sodass am Ende der Zeile Platz für weitere Noten entsteht. Die Noten, die am Anfang der Zeile stehen, werden hierbei nach links aus der Anzeige „herausgeschoben“ und fortan nicht mehr angezeigt.

Bestätigen Sie Ihre Einstellungen abschließend durch Drücken des „OK“-Knopfes. Nun erscheint im Bildschirm rechts neben dem Fenster mit der Benutzeroberfläche ein weiteres Fenster (mit Notenlinien und dem gewählten Notenschlüssel und Takt). Damit ist die Konfiguration abgeschlossen und Sie können ab sofort Noten mit Hilfe des MIDI-Keyboards eingeben. Diese werden im rechten Fenster entsprechend angezeigt.
Je nachdem welchen Notenschlüssel Sie gewählt haben, können unterschiedliche Notenbereiche dargestellt werden. Der Violinschlüssel umfasst hier die Töne zwischen f und e”’, der Bassschlüssel die Töne zwischen A, und g’. Wenn Sie Töne eingeben, die außerhalb des momentan gültigen Bereichs liegen, so werden diese ignoriert.

Bitte bestätigen Sie nachträgliche Änderungen an den Einstellungen ebenfalls durch Betätigen des „OK“-Knopfes. Beachten Sie bitte, dass bei der nachträglichen Änderungen des Notenschlüssels die Anzeige der Noten zurückgesetzt wird.

Die beiden Knöpfe („ZURUECK“ und „RESET“), die während der Konfiguration noch keine Funktion hatten, können jetzt verwendet werden. Der „ZURUECK“-Knopf löscht hierbei die **zuletzt** eingegebene Note. Wurde zuletzt ein Akkord eigegeben (gilt nur für den Tonfolgemodus), wird dieser als Ganzes gelöscht. Der „RESET“-Knopf setzt die gesamte Anzeige zurück, d.h. er löscht **alle** eingegebenen Noten. Durch das Schließen von einem der beiden Fenster können Sie die Anwendung beenden.

Nun können Sie mit der Bewertung der Anwendung beginnen. Fühlen Sie sich frei mit den verschiedenen Funktionen der Anwendung zu experimentieren und die verschiedenen Möglichkeiten auszutesten. Gehen Sie dabei gerne so vor, wie Sie es in einer normalen Unterrichtssituation auch tun würden. Falls dabei Fragen oder Probleme auftreten, wenden Sie sich bitte an mich.

(HINWEIS: Die Anwendung unterstützt aktuell noch keine Soundausgabe)

Fragebogen

UNIVERSITÄT KOBLENZ · LANDAU

Auf den folgenden Seiten finden Sie einige Fragen, die der Bewertung der Software im Hinblick auf die Erfüllung der aufgestellten Anforderungen und auf die Tauglichkeit für den geplanten Einsatz dienen. Bei den meisten Fragen geht es darum die genannten Aspekte anhand einer gegebenen Skala zu beurteilen. Außerdem stehen Ihnen sowohl einzelne Kommentarfeldern am Ende eines jeden Fragenblocks als auch mehrere Textfelder ganz am Ende des Fragebogens zur Verfügung, um positive sowie negative Bemerkungen zu hinterlassen. Die einzelnen Fragentypen sind weitestgehend selbsterklärend, werden Ihnen aber bei Bedarf im Laufe des Fragebogens zusätzlich kurz erklärt.

Erwartungen

1. Was erwarten Sie von der Anwendung?

.....

.....

.....

.....

.....

MIDI-Eingabe

Bitte bewerten Sie die folgenden Aspekte auf einer Skala von 1 - 5.
1 ist sehr schlecht, 5 sehr gut.

2. Wie intuitiv konnten Sie die Noten mit Hilfe des MIDI-Keyboard eingeben?

sehr schlecht				**sehr gut**
☐	☐	☐	☐	☐
1	2	3	4	5

Kommentar:

.....

.....

.....

.....

.....

Fragebogen

UNIVERSITÄT KOBLENZ · LANDAU

Optik

Bitte bewerten Sie die folgenden Aspekte auf einer Skala von 1 - 5.
1 ist sehr schlecht, 5 sehr gut.

3. Empfanden Sie die Optik der Notenschrift als ansprechend?

überhaupt nicht				**voll ganz**	**und**
☐	☐	☐	☐	☐	
1	2	3	4	5	

4. Empfanden Sie die Optik der Notenlinien als ansprechend?

überhaupt nicht				**voll ganz**	**und**
☐	☐	☐	☐	☐	
1	2	3	4	5	

Darstellungsmodi

Bitte bewerten Sie die folgenden Aspekte auf einer Skala von 1 - 5.
1 ist sehr schlecht, 5 sehr gut

5. Wie haben Ihnen die verschiedenen Darstellungsmodi gefallen?

	1	**2**	**3**	**4**	**5**
Tonfolgemodus	☐	☐	☐	☐	☐
Akkordmodus	☐	☐	☐	☐	☐

	1	**2**	**3**	**4**	**5**
Notenblattmodus	☐	☐	☐	☐	☐
Durchlaufmodus	☐	☐	☐	☐	☐

6. Hat das Wechseln der Darstellungsmodi flüssig funktioniert?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

7. Welchen Darstellungsmodus empfanden Sie als am besten/hilfreichsten? Wählen Sie aus beiden Gruppen je einen Modus aus.

- ☐ Tonfolgemodus
- ☐ Akkordmodus
- ☐ Notenblattmodus
- ☐ Durchlaufmodus

Genauigkeit/Korrektheit

Bitte bewerten Sie die folgenden Aspekte auf einer Skala von 1 - 5.
1 ist sehr schlecht, 5 sehr gut

8. Bewerten Sie die Korrektheit der Darstellung der folgenden Elemente:

	voll-ständig falsch **1**	über-wiegend falsch **2**	halb und halb **3**	über-wiegend korrekt **4**	**voll-ständig korrekt** **5**
Tonhöhen	☐	☐	☐	☐	☐
Versetzungszeichen	☐	☐	☐	☐	☐
Tonlängen	☐	☐	☐	☐	☐
Taktstriche	☐	☐	☐	☐	☐
Hilfslinien	☐	☐	☐	☐	☐
Akkorde	☐	☐	☐	☐	☐

Kommentar:

.....

.....

.....

.....

.....

Performanz

Bitte bewerten Sie die folgenden Aspekte auf einer Skala von 1 - 5.
1 ist sehr schlecht, 5 sehr gut

9. Lief die Anwendung flüssig?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

10. Wurden alle Noten, die Sie eingegeben haben, auch angezeigt?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

11. Hat die Bildschirmausgabe angemessen schnell auf Ihre Eingaben mit dem Keyboard reagiert?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

Kommentar:

.....

.....

.....

.....

.....

Fragebogen

UNIVERSITÄT KOBLENZ · LANDAU

Einstellungen

Bitte bewerten Sie die folgenden Aspekte auf einer Skala von 1 - 5.
1 ist sehr schlecht, 5 sehr gut

12. Empfanden Sie die Optik der Benutzerschnittstelle als ansprechend?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

13. Empfanden Sie die Benutzerschnittstelle als übersichtlich?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

14. War die Bedienung der Benutzerschnittstelle einfach und intuitiv?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

15. Wie war Ihr Gesamteindruck der Benutzerschnittstelle?

sehr schlecht				**sehr gut**
☐	☐	☐	☐	☐
1	2	3	4	5

Kommentar

.....

.....

.....

.....

.....

Fragebogen

UNIVERSITÄT KOBLENZ · LANDAU

Gesamteindruck der Anwendung

Bitte bewerten Sie die folgenden Aspekte auf einer Skala von 1 - 5.
1 ist sehr schlecht, 5 sehr gut

16. Wird Ihnen die Anwendung bei der Unterrichtsgestaltung von Nutzen sein?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

17. Wie zufrieden waren Sie mit der Anwendung insgesamt?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

18. Wurden Ihre Erwartungen an die Anwendung erfüllt?

überhaupt nicht				**voll und ganz**
☐	☐	☐	☐	☐
1	2	3	4	5

19. Im Folgenden sind einige Funktionen genannt, die zur Anwendung hinzugefügt werden könnten. **Bewerten Sie wie wichtig Ihnen die einzelnen Aspekte sind, indem Sie durch Eintragung der Zahlen 1-10 in die Kästchen eine Rangfolge erstellen.** Es stehen Ihnen 4 freie Plätze zur Verfügung, um eigene Ideen einzutragen und ebenfalls in die Rangfolge aufzunehmen. Es müssen nicht alle Funktionen eingeordnet werden.

Funktion	**Wichtigkeit**
Pausen	[ ]
Soundausgabe	[ ]
Einstellung der Tonart	[ ]
Speichern&Laden	[ ]
Einstellung fester Notenwerte	[ ]
Notendarstellung im Klaviersystem	[ ]
_____	[ ]
_____	[ ]
_____	[ ]
_____	[ ]

20. Was hat Ihnen besonders gut gefallen?

.....

.....

.....

.....

.....

.....

.....

21. Was hat Ihnen gar nicht gefallen?

.....

.....

.....

.....

.....

.....

.....

Vielen Dank für Ihre Teilnahme an der Evaluation!