Allegheny College
Department of Computer Science
&
Department of Music

Senior Thesis

Computer Aided Music Composition

by

Jacob Sutter

ALLEGHENY COLLEGE

COMPUTER SCIENCE
&
MUSIC

Project Supervisors: **Dr. Janyl Jumadinova & Dr. Douglas Jurs**
Co-Supervisors: **Dr. Douglas Lumen & Dr. James Niblock**

April 11, 2020

Abstract

This project set out to create a tool that would allow a user, with any level of musical or computer science knowledge, to compose a melody. The tool contains a learning and feedback system as part of a user interface to guide the user through the composition process. This document details the background, history, design, creation, testing, and results of this tool. The history and background are provided through existing AI powered composers. The design and creation of this tool is discussed in detail with source code examples and explanations. The testing and results of this project were generated through a user experience study around the reactions to the tool and its user interface.

Acknowledgments

- Dr. Douglas Jurs and Dr. Janyl Jumadinova for help in designing the research
- Dr. Lowell Hepler and Dr. James Niblock for help in executing the research
- Dr. James Niblock for help in proofing and editing the manuscript

To Barbara Moore

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The involvement of computers in the creation, performance, distribution, and analysis of music is a phenomenon that most everyone is aware of regardless of their experience in using these technologies. Since the creation of the computer, musicians, researchers, and creative people alike have found new and innovative ways to integrate technology into the field of music.

Some of these innovations, that were once state of the art, now seem common place include digital audio recording, synthesizers, and music notation software. As we further advance computers and digital music, we will create and discover new methods to integrate technology into music and find ways to improve our existing means of integration.

Currently, in the field of computer science, artificial intelligence is being heavily developed and researched. Its uses, implications, practicality, and future are an important topic of discussion in our lives today. Specifically related to the uses of AI, artists and musicians are examining ways in which it can be applied to their own processes.

When the idea of computers creating art and music with computers was initially conceived, the field of AI was nonexistent. Today, the development of AI has improved extensively, but does not come without limitations. However, it is now possible to train computers to create and develop original works of both music and art.

A parallel idea would then be to use computers to teach humans music. This project set out to try to accomplish this task by creating a computer system that is capable of assisting a person in the creation and validation of a melody. It was designed in such a way to allow a person with any degree of music or computer knowledge to be able to compose a melody and receive feedback and instruction for how to improve it.

1.1 Background

The earliest cited example of a musical composition that was created using a computer was Hiller and Isaacson's *Illiac Suite* [5]. This work was composed in 1956 by Lejaren

Hiller and Leonard Isaacson [4]. The following diagram shows the rules that were programmed into the system to guide the composition [7]. The system of rules that they developed were meant to adhere to the format of strict counterpoint from the Baroque period [7]. These rules cover cadences, resolutions, voice leading, and voice spacing.

Figure 1.1: Diagram from Hiller and Isaacson's published paper on the Illiac Suite showing counterpoint rules [7]

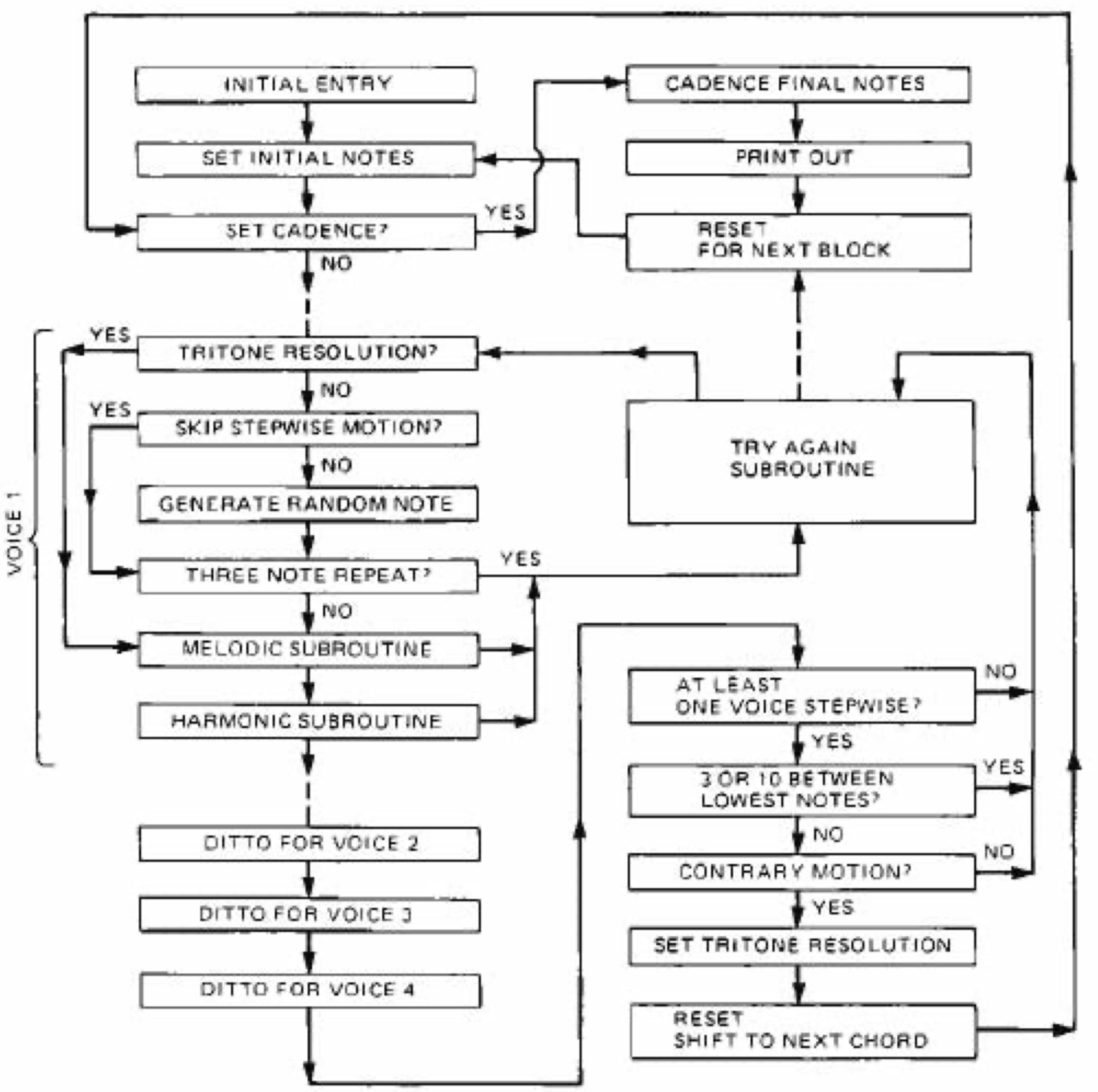


Figure 1.1
Experiment 2: Main routine for strict counterpoint.

An example of a later approach to computer music composition was Hiller and John Cage's HPSCHD in 1968 [4]. This work was composed by using computers that played Mozart's Musical Dice Game and altered the results using the rules of the Chinese oracle, I Ching [4].

The development of each of these works required extensive music composition and computer programming knowledge to complete. Lejaren Hiller was a composer by trade and Leonard Isaacson was a professor of computer science [4]. Neither of these qualifications are applicable to describe the general population of people that are interested in music. Music has an infinitely deep reaching appeal to practically all humans and even many animals.

With that being said, it only makes sense then that people who are not educated in either music or computer science should be provided with a way to create music. By not providing the means for these people to express themselves musically, we miss out on a large amount of creative output and new musical compositions. Rather than requiring these people to try to gain access to training, we can assist them in musical composition with computer based technology.

1.2 Motivation

Imagine that the piano was an instrument that could only be played if you were over seven feet tall. If the instrument were designed like this, only a very small portion of the population would be able to make use of it. The piano, however, is one of the most universally played instruments because it was not designed in a way that excludes lots of people from using it.

Currently, the tools that exist for computer assisted music composition are only accessible to musicians with high level computer science training [15]. All of them either have generally inaccessible command line interfaces or require the user to be trained in music composition to be able to understand or use the output of the program. Neither of these situations cater to the general person who might find some interest in being able to express themselves by composing with these tools.

Due to the research heavy focus around the creation of these tools, not a single one was designed to be placed in the hands of the average user. Someone who may have always wanted to create music might not have access to the training, schooling, tools, or opportunities that are currently required to be able to compose, record, or play music. Giving these people a tool that can help them through the composition process and assist them in notating their music is an important first step in being able to make creating music more accessible to the average person.

1.3 Goals of the Project

The goal of this project was to create a computer based musical composition tool that can be used by any person and does not require music or computer science knowledge. Through the careful design and testing of a user interface built over the high level functions of the composer, this project set out to create a tool that makes music composition accessible to everyone.

In order to make this a reality, the project was broken into three main components.

1. Computer Composer Development
2. User Interface Development
3. User Experience Study

1.3.1 Computer Composer Development

This part of the project consisted of developing the backend functions of the computer composer. Several different preexisting tools were combined to create the composer and to process its output. The basic primary functions of the composer include melody generation, notation, playback, analysis, and output.

Each of these is a necessary part of the tool in order to make the composer into one that covers all of the bases of musical composition. After the generation of the musical output, the user is free to do what they would like in changing and altering it either with the composer or externally.

1.3.2 User Interface Development

This next part of the project consisted of building the interface that the user would use to interact with the composer. There are already several tools that perform some combination of the functions of the composer that were listed above, but none of them have this interface component.

In order to meet the overall goal and motivation for this project, the interface had to be designed in a way that could, in essence, explain itself. Rather than having long lists of instructions and using advanced music theory terminology, the interface has to present itself in a way that each element explains its own function in a completely visual way. This is where the bulk of the work of the project was done.

1.3.3 User Experience Study

Following the development of the composer and the UI, it was necessary to test the effectiveness of the UI and if it met the idea behind the project through a user study. Users with varying degrees of musical knowledge were asked about their experiences through interacting with the tool. The overall success of the interface design was gauged through a series of survey questions.

After getting feedback from users of the tool, some revisions were performed to the interface to clear up uncertainties and confusions that the users had. This is an important step in UI development since not every user of the interface is going to see it or interact with it in the same way.

1.4 Thesis Outline

The following outlines what is contained within this thesis document.

1.4.1 Introduction

This section of the document situates the work within the disciplines of music and computer science. It contains details about the history of computer composed music and some background into the associated problems. It also discusses the motivation for the project and its overall goals.

1.4.2 Related Work

This section of the thesis document provides a discussion of other published works related to computer composed music. It discusses the details and ideas used in these projects as well as some of their shortcomings as related to the goals of this project.

1.4.3 Method of Approach

This section details the approach taken to solve the problem discussed during the introduction. It explains the composer and user interface development. It further discusses the reasons for these parts of the project and what was entailed in their completion.

1.4.4 Experimental Results

This section of the document provides a discussion of the user study and the overall successes and failures of the project. It summarizes the users' responses to the questions in the study and what their responses mean for the evaluation of the system.

1.4.5 Discussion and Future Work

This section of the thesis document discusses the overall impact of this project and what might be done in the future to improve it. Neither of these are limited to strictly the research-based impact of this project as the goal of this project is to move the tool beyond the lab and into the hands of the average user.

Chapter 2

Related Work

As the field of AI has grown, more and more researchers are looking into the extent of the capabilities of AI. The creation of music with computers has gained increased attention within the last decade, however, this does not mean that these tools have gained popularity among users. Quite a few of these composers have been developed, but only a very small number of them have seen industry use and almost none of them have been used outside of research or these niche areas in industry.

The reason for this primarily is that the focus of researchers when creating these tools is how advanced the composer can be and what new ways can AI be employed to create more original music. Their focus is not at all on how these tools can be made in a way that they might find a larger user base or even a user base at all. This limits what might come of these technologies now and in the future.

Setting this aside, however, the researchers that have developed these composers have created some very impressive tools capable of creating very convincing music. This project aims more to guide the user through the process of composing music. This is different than a majority of the research projects in this area that focus on creating a composer that can write its own music, but the writing still contains an important discussion of how computers can be used to create music.

2.1 AI Powered Composers

The following works discuss several different AI based composers that were created to generate their own music. For the purpose of this thesis, they will be analyzed for their effectiveness at composing as well as their shortcomings as tools that only have limited usability. All of these have proven to be effective at creating music, but none of them have shown popularity as music composition tools.

2.1.1 FlowComposer

Of the existing AI composers, FlowComposer is the only one that has seen use in industry [11]. An album that was recorded consisting of only music created with

FlowComposer was released in 2017 [11]. This is also one of the only tools that has an actual visual user interface and is not strictly accessed through the command line. It is currently being developed as part of Sony CSL and Flow Machines, but is not available to the public for use [11]. An audio example of what FlowComposer is capable of producing is available on their website, `https://www.flow-machines.com`, but access to the tool itself is not provided [13].

The French composer Jean-Michel Jarre is using Flow Machines to create an "infinite album" [12]. This is a never ending, constantly evolving piece of music generated by Flow Machines from a large library of musical samples [12]. The album is available in JarreLab's EON app [8]. The music that you hear is generated in real time and will never be heard again [8]. A link to download the app can be found here: `https://jeanmicheljarre.com`.

FlowComposer was designed to create new songs automatically in any style or can generate the style based on user provided parameters and input [11]. This tool is capable of re-harmonization (taking an existing melody and create new harmonies in different styles), variation (taking the melody and introducing variations into the pitch content and rhythm), and rendering (playing back a given score as if it were being performed) [13].

The musical output of this tool is a lead sheet that is scored for a full band. The music that FlowComposer produces fits into the category of popular songs (e.g. Pop, Jazz, Rock, R & B, etc.). It comes out fully notated and able to be performed or recorded. It is a very high quality tool and very capable at writing songs, but due to its research based nature, is kept from the public. It is highly likely also that this tool would be very expensive if it were available to the public. Maybe in the future, this will be a tool that is made available, but it will doubtfully ever be accessible to the average person due to the potential price.

2.1.2 BachProp

BachProp is a neural composer algorithm that was created to be able to compose new music in any style [2]. It was trained initially on the chorales of J. S. Bach, but is able to compose in any style when given appropriate training data [2]. The data is fed to the composer through MIDI files which are mathematically normalized before being translated into probability data that the algorithm uses to predict melodic direction and shape [2].

For the evaluation of BachProp, audiences were asked to rate several string quartets that it composed after it was trained on string quartets by Haydn and Mozart [2]. Based on the results of the surveys, the music was not only well received, but it was also quite convincingly in the appropriate style and character [2]. To hear some of the music that Bach Prop composed, visit: `https://sites.google.com/view/bachprop-icml18/`.

The process of actually performing crowd based musical validation is quite important here. Rather than the researchers who developed the composer rating the music, they had general audiences perform the reviews to better understand the broad appeal of the music. In this project, a similar crowd based reviewing of the interface was performed. Participants individually judged their created music in their own terms so that their personal success was measured to be able to determine if the user interface was able to guide them through the composition process.

BachProp has also been used and evaluated in the composition of music in a number of other styles [1]. The method of musical representation used by BachProp has proven to be more effective at capturing and later translating the information from the provided training scores than other algorithmic composers [1]. By normalizing how the musical data is stored, this tool is able to gather and retain more detail from the score than other composers [1]. This is what makes BachProp more effective at emulating a given style and producing more convincing musical output [1].

In order for the several Python libraries that were used to create the composer to cooperate, the musical data for this project was also represented in a normalized manner. Like BachProp, the data was represented as MIDI events to provide a standard for communication and output. This also means that at any step in the composition process the data could be imported into any standard music notation program and used externally to the composer as MIDI is the standard for digital music notation representation and playback.

The largest shortcoming of BachProp is that it was not designed in a way that can be used without extensive music or computer science knowledge. This means that is highly unlikely that it would ever be picked up by someone who simply wanted to play around with it or experiment with it. There is an extensive setup involved and it cannot be used by any simple means. However, BachProp is able to produce high quality music in any style and not just popular music as FlowComposer does.

2.1.3 DeepBach

DeepBach is a graphical model for musical composition that is designed to be effective at producing polyphonic music in four part hymn format [6]. This is yet another composer that was trained on the music of J. S. Bach in order to teach it the mechanics of composition and in this case, to teach it chorales specifically [6]. This composer is able to produce convincing Bach chorale style pieces that are driven by user parameters and input [6]. The following are examples of chorals composed by DeepBach. Below each, is a chordal analysis in slash chord notation.

It is important to note that examples (a) and (b) were created using the same user input data and parameters. This shows that there are at least several different possibilities for harmonization and development of the melodic line. Before the individual examples are discussed, there are some important similarities that are shared between the chorales.

The first of these is the handling of steps and leaps in each of the voices. It will later be discussed that it is appropriate to move in step wise motion in the opposite direction after leaping. In each voice in each example this rule is exactly followed.

The next similarity is in the cadences at the ends of the chorales. In both example (a) and (c) we see, in the last three chords, a move from some version of V7 of V, to V, to I. This is a very typical progression and is resolved correctly in both the soprano and bass voices by stepping and leaping respectively. In example (b) we have a similar progression, but it does not use V7 of V.

Another similarity comes in the treatment of the different voices. The soprano voice, which has the melody, contains much less motion and overall activity as compared to the other voices. This is good practice in melody composition as the melody should be easy to remember and sing back.

Yet another similarity can be found with the use of chromaticism in the passing tones and shift between chords. To provide both harmonic interest and a form of melodic interest within each of the parts, come chromatic motion is used. When it is used, it is resolved appropriately with stepwise motion.

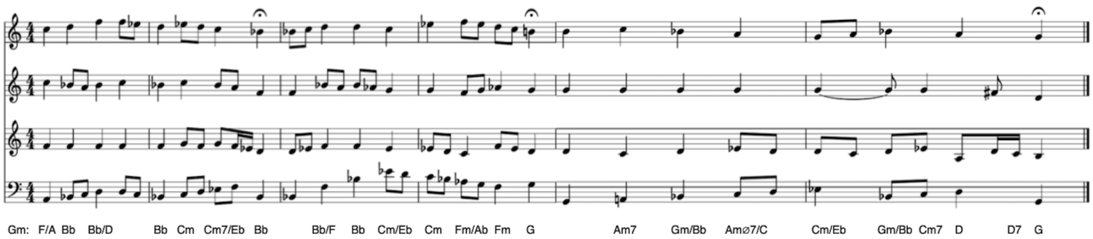
Figure 2.1: Score (a) from Hadjeres's published paper showing DeepBach output [6]

(a)

This example (a) is in the key of G Minor. While it does end in G Major, the presence of B♭'s and E♭'s put it in G Minor. The fact that it ends in G Major is not uncommon however. It was often the practice to end a chorale in a minor key with the major version of the I chord by raising the third.

This example in particular uses more chords outside of the normal key material than the other two. However, the two cadences within the chorale (indicated by the fermatas) are both firmly within G Minor.

Figure 2.2: Score (b) from Hadjeres's published paper showing DeepBach output [6]



(b)

As mentioned previously, this example (b) was made with the same user input data as the previous example. Also like the first example, this one also ends in the parallel major. This is not a requirement of this style. Both examples just happen to leverage this technique.

This example has some out of key material, but not as much as the first example. It also moves to the relative major in the second of the two internal cadences instead of only at the end. While they do share the same metadata, they are distinct in their melodies, progressions, and voice leading.

Figure 2.3: Score (c) from Hadjeres's published paper showing DeepBach output [6]

(c)

This third example (c) is in C Major. It also has one more cadence than the other two examples. Another factor that makes this example different is that there is noticeably more chromatic motion in each of the voice parts.

While this has more material from outside the key than the second example, it does not have as much as the first. In this case however, the internal cadences are from outside the key. They are, however, distantly related as E Major is V of A Major, A Major is V of D Major, D Major is V of G Major, and G Major is V of C Major. This progression can be seen across the chords at the second two fermatas and final three chords.

DeepBach does provide integration with the notation program MuseScore [6]. This means that there is a way in which people who are familiar with notation software could use this composer. While this is better than many of the others, it still requires a fair bit of prior knowledge. Additionally, this composer is not capable of operating in more than four part chorale style. The potential uses of this composer are very limited due to this fact.

Chapter 3

Method of Approach

This section has been broken down into two parts that each detail how each of the main composer components were developed. The first section describes the processes and tools needed to develop the composer and the second section describes the processes and ideas behind the development of the user interface.

3.1 Computer Composer Development

To begin this project, it was necessary to develop the composer which would function as the backend of the tool. This was chosen as the first step because the elements of the user interface would have to be chosen and designed based on what the composer was capable of doing. Without first implementing the basic functionalities of the composer, it would be impossible to determine how the user interface should be built. To develop the functions of the composer, several preexisting python tools and libraries were employed.

3.1.1 Tools

The following table lists all of the tools that were used and their specific functions within the composer. None of these tools, on their own, where capable of forming the functioning composer. Several of these tools have very advanced sets of functions, but no single tool had all of the required functions for the composer.

Table 3.1: Tools and packages and their uses in this project

LilyPond	Notation
mingus	MIDI File Generation
MIDIjs	Playback
music21	Melody Analysis

LilyPond

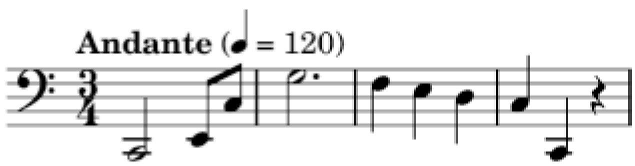
The primary function of LilyPond is as a text-based music engraving tool [9]. LilyPond can either be used with a graphical user interface or as a command line tool by interfacing with another program [9]. For the purpose of this tool, the command line version is being used by interfacing with mingus.

This tool was chosen for this project due to its ease of integration and because of its ability to output high quality notation. Additionally, the notation generated by LilyPond is easily customized and extremely flexible. There are many options for editing the layout and spacing of the notation easily within the command line.

To use LilyPond with the command line, the user must generate a text file. The tools then uses the data in this file to draw the notation in the desired format. For this tool, the text file is generated in the background as the user chooses notes and durations. Then, the notation is drawn into an image that is displayed on the page. A sample of the text file and the associated output is shown below.

Figure 3.1: LilyPond Text File and Output Example [9]

```
\relative {
  \clef "bass"
  \time 3/4
  \tempo "Andante" 4 = 120
  c,2 e8 c'
  g'2.
  f4 e d
  c4 c, r
}
```



The `\relative` in the text file has to deal with octave designation. Adding `\relative` sets each previous note as the center pitch and determines the octave of the following note to be the closest instance of that pitch either up or down. Adding the `'` specifies how many octaves higher the note should be from the center pitch and adding the `,` specifies how many octaves lower it should be from the center pitch. The number of `'`'s or `,`'s added is the number of octaves to shift the pitch. The `\clef`, `\time`, and `\tempo` indications refer to exactly for which they are named.

In terms of pitch and duration content, the pitch is given by the letter name and duration is given by the number. `1` for whole note, `2` for half note, `4` for quarter note, etc. The `.` specifies whether or not that duration is meant to be dotted. Finally, the `r` specifies a rest.

Also important to note, the spacing within the lines of the text file and the newlines themselves are not taken into account by LilyPond. These are simply for readability and are not required to generate the notation. It appears as though the barlines may have been created this way, but are actually added automatically based on the meter signature.

mingus

mingus is a package for Python that provides advanced music theory and notation functions [14]. For this project, it is being used to generate MIDI files to be used later in playback and to interface with LilyPond to generate music notation.

This tool was chosen for this project due to its ease of use and flexibility when generating files. It has many functions that make it useful that are beyond the scope of this project, but in terms of creating MIDI files, it has the most direct method of doing this among similar tools.

In order to generate a MIDI file, a NoteContainer must be created first. This is quite simple to do in mingus. After this step, there is a function that turns the NoteContainer into a MIDI file. An example of this is shown below.

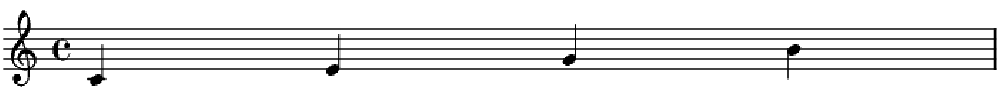
Figure 3.2: mingus MIDI File Generation [14]

```
> nc = NoteContainer(["A", "C", "E"])
> MidiFileOut.write_NoteContainer("test.mid", nc)
```

Along with creating the MIDI files for playback, mingus also handles the interface with LilyPond in order to generate music notation. The following example shows the process of building a bar in music with specific notes and then converting that bar into an image file that contains the notation.

Figure 3.3: mingus LilyPond Notation Image File Generation [14]

```
> b = Bar()
> b + "C"
> b + "E"
> b + "G"
> b + "B"
> bar = LilyPond.from_Bar(b)
> LilyPond.to_png(bar, "my_first_bar")
```



MIDIjs

MIDIjs is a JavaScipt MIDI player that works with all modern browsers and is built entirely with JavaScript [10]. This tools was chosen for this project due to its high sound quality and ability to run without requiring any sort of download. It runs in place and requires no plugins or extensions [10]. The following example shows how to play a MIDI file using MIDIjs.

Figure 3.4: MIDIjs MIDI File Playback [10]

```
<script type='text/javascript'src='//www.midijs.net/lib/midi.js'></script>
<a href="# " onClick="MIDIjs.play('file.mid');">Play file.mid</a>
<a href="# " onClick="MIDIjs.stop();">Stop MIDI Playback</a>
```

music21

music21 is a Python tool for computer analysis of music and musical data [3]. It is capable of analysis of large and small works alike with analytical categories such as harmony, form, structure, pitch content, rhythmic content, lyric content, frequency, intervals, and couterpoint. It is capable of performing roman numeral analysis and generating post-tonal matrices. In this project, music 21 is responsible for melodic analysis and providing the data to display feedback to the user on how to improve their melody.

music21 was chosen for this project due to its deep and extensive analytical tools. It is capable of powering all of the chosen elements of melodic analysis to assist the user during composition. Several of the functions that are used within the feedback system are demonstrated below.

This first example demonstrates how music21 is capable of checking for consonances and dissonances between pitches. A starting and ending pitch are given and the function returns true if the interval is consonant and false if it is dissonant. Intervals that are considered consonant by music21 are major or minor thirds or sixths, perfect fifths, and unisons.

Figure 3.5: music21 Consonant Interval Checking [3]

```
> i1 = interval.Interval(note.Note('C'), note.Note('E'))
> i1.isConsonant()
True
> i1 = interval.Interval(note.Note('B-'), note.Note('C'))
> i1.isConsonant()
False
```

This next example shows how music21 can identify the interval between two pitches. The starting and ending notes are specified and a function is used to display the interval between them. If `simpleName` is used, the interval is reduced to less than an octave. If `semiSimpleName` is used, the interval is reduced to no more than an octave.

Figure 3.6: music21 Interval Identification [3]

```
> n1 = note.Note('c3')
> n2 = note.Note('c5')
> aInterval = interval.Interval(noteStart=n1, noteEnd=n2)
> aInterval.name
'P15'
> aInterval.simpleName
'P1'
> aInterval.semiSimpleName
'P8'
```

This last example demonstrates how music21 can provide a pitch based on a starting pitch and an interval. Once the starting pitch and the interval are given, a function can be used to return a new pitch based on a transposition of the first pitch.

Figure 3.7: music21 Find Second Note Based on Interval [3]

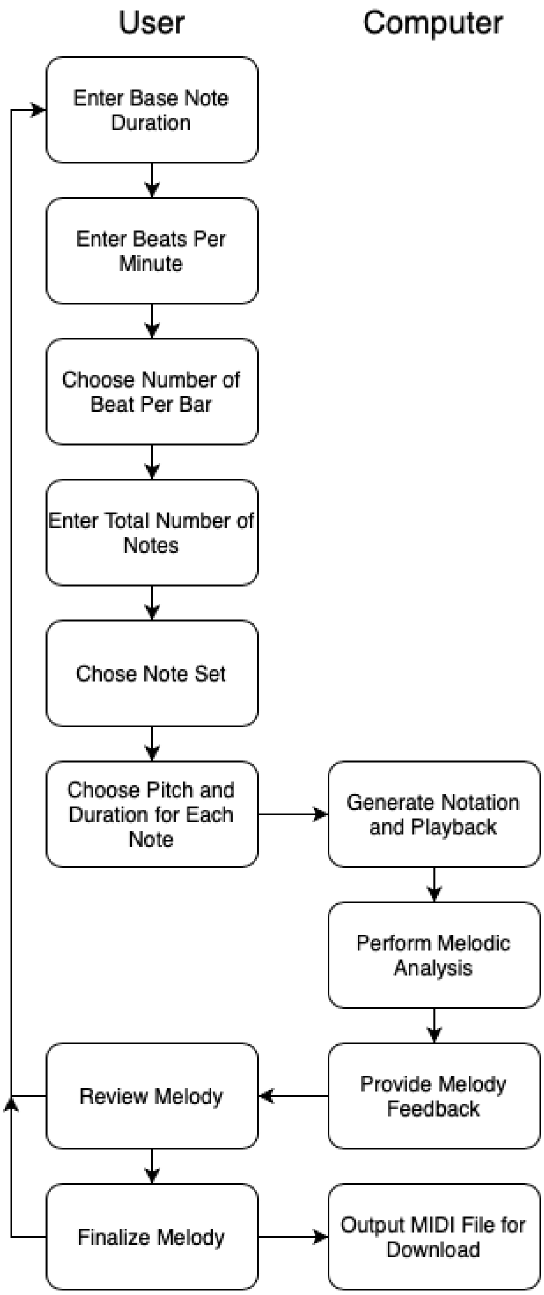
```
> p1 = pitch.Pitch('A# 4')
> i = interval.Interval('m3')
> p2 = i.transposePitch(p1)
> p2
<music21.pitch.Pitch C# 5>
```

Later in the document, it will be shown how each of these functions were used to generate recommendations for how the user can improve their melody.

3.1.2 Main Composer Process

The following diagram shows the composition process broken down into the specific actions of the computer and the user. While the user is responsible for more actions than the computer, the user is assisted in the process by the specific design of the interface and the directions that are provided when using the composer. The computer then assists the user with melodic analysis to verify the quality of the melody in practical music theory terms.

Figure 3.8: Composition Process Flowchart



There are several important attributes that the user must choose when first creating their melody. The following table lists each of these elements and their definitions in the context of this tool. It is important to note that none of these values are final and they can be changed at any time.

Table 3.2: Melody Initialization Attributes

Base Note Duration	Base unit for time measurement and division
Beats Per Minute	Number of base units per minute
Beats Per Bar	Number of base units in each measure of music
Total Number of Notes	Number of notes in the melody
Note Set	Set of pitches available within the user's chosen key

Base Note Duration

In the context of this tool, this base note duration functions as the beat unit. In music theory, the beat unit is defined as the note duration that gets on beat. This is the bottom number in a meter signature. The user has the option to select 2, 4, 8, or 16 as the base duration.

Beats Per Minute

In the context of this tool, tempo is represented as the number of beats that occur per minute. This aligns with the standard definition of tempo as the rate at which beats pass. The duration that defines the beat in this tool is what the user selected as the base note duration.

Beats Per Bar

To finish out the creation of the meter signature, the user must select the number of beats that will occur in each measure of music. The user is not limited to any specific values here. They can have as many or as few beats per bar as they desire. In terms of the meter signature, this is the top number.

Total Number of Notes

Before the user begins to input notes, they must choose how many note entry forms that they want to render into the interface. This number can be chosen arbitrarily as there is a good chance that they will not know the exact number that they want. If this value is changed, the extra notes will be removed from the end.

Note Set

In the context of this tool, the note set is effectively the key signature. The user will initially choose a starting pitch and a mode. This will then define the set of notes that are available for the user to choose within the composer. By limiting the notes to a particular key, this helps the user to be able to compose more coherent melodies. It is also possible to manually add an accidental to any note. The available accidentals are double flat 𝄫, flat ♭, natural ♮, sharp ♯, and double sharp 𝄪. The following note sets are available within the composer:

Table 3.3: Available Note Sets

C♭Major	C♭D♭E♭F♭G♭A♭B♭
G♭Major	G♭A♭B♭C♭D♭E♭F
D♭Major	D♭E♭F G♭A♭B♭C
A♭Major	A♭B♭C D♭E♭F G
E♭Major	E♭F G A♭B♭C D
B♭Major	B♭C D E♭F G A
F Major	F G A B♭C D E
C Major	C D E F G A B
G Major	G A B C D E F♯
D Major	D E F♯G A B C♯
A Major	A B C♯D E F♯G♯
E Major	E F♯G♯A B C♯D♯
B Major	B C♯D♯E F♯G♯A♯
F♯Major	F♯G♯A♯B C♯D♯E♯
C♯Major	C♯D♯E♯F♯G♯A♯B♯
A♯Minor	A♯B♯C♯D♯E♯F♯G♯
D♯Minor	D♯E♯F♯G♯A♯B C♯
G♯Minor	G♯A♯B C♯D♯E F♯
C♯Minor	C♯D♯E F♯G♯A B
F♯Minor	F♯G♯A B C♯D E
B Minor	B C♯D E F♯G A
E Minor	E F♯G A B C D
A Minor	A B C D E F G
D Minor	D E F G A B♭C
G Minor	G A B♭C D E♭F
C Minor	C D E♭F G A♭B♭
F Minor	F G A♭B♭C D♭E♭
B♭Minor	B♭C D♭E♭F G♭A♭
E♭Minor	E♭F G♭A♭B♭C♭D♭
A♭Minor	A♭B♭C♭D♭E♭F♭G♭

Durations

Once this initial setup is complete, the user can begin to enter the pitches and durations for each of the notes in the melody. The pitches that are available to the user are a three octave span of the pitch classes in the key (note set) that the user chose. The durations and the the decimal value that the user will see are listed in the following table. The equivalent rest is available for each duration when no pitch is selected.

Table 3.4: Available Duration Values

Whole Note	𝅝	4
Dotted Half Note	𝅗𝅥.	3
Half Note	𝅗𝅥	2
Dotted Quarter Note	♩.	1.5
Quarter Note	♩	1
Dotted Eighth Note	♪.	0.75
Eighth Note	♪	0.5
Dotted Sixteenth Note	𝅘𝅥𝅯.	0.375
Sixteenth Note	𝅘𝅥𝅯	0.25

Notation and Playback

Once the user enters some pitches and duration values, they are then able to generate a snippet of musical notation and hear what they have written. For this part of the composer, two of the other preexisting Python tools come into play. mingus and LilyPond handle the notation and MIDIjs handles the playback within the composer. It is possible to export the raw MIDI files and open it within external notation software, but this will be discussed later.

Once the user begins to input duration values and pitches, it is possible to generate playback. To do this, the data that the user enters is translated into a form that can be read by mingus. The functions within mingus to generate MIDI files and to use LilyPond to generate an image file with the notation are then called. The MIDI file is then sent to MIDIjs for playback and the image created by LilyPond is displayed to the user.

Melodic Analysis

It is during this part of the process that the system generates feedback on the user's melody and provides suggestions for how to improve it. The following list details each of the areas of feedback and the specific functions within music21 that will handle the checks. These rules for "good" melodies are based in western music. By referring to these rules as "what makes a good melody" it is simply to say that these are some of the properties of melodies in western music that have proven to make melodies that are pleasing to hear and easy to perform. This is not to say that melodies that do not follow these rules are bad. These are just some guidelines to help those that are new to melody creation.

- Range
  - Typically, the range from the lowest note to the highest note should be under an octave and a half. This is to make sure that the melody is within a range that most people and instruments can produce good sound.

```
> analysis.discrete.Ambitus().getPitchRanges(input)
```

- This function returns the difference between the lowest and highest note by calculating the pitch space. Conditional statements are then used to determine if this is within the appropriate range.

- Contour
  - The melody should consist mostly of stepwise motion. This means that there should be primarily movement by whole and half steps (conjunct motion) with some leaps (disjunct motion) added for interest. Generally, leaps should be used to outline a climax in the phrase and work towards the resolution with stepwise motion. It is also a good idea to keep the size of the leap no more than an octave.

```
> voiceLeading.NNoteLinearSegment(listOfNotes).melodicIntervals
```

  - This function returns a list of the intervals between each consecutive note in the melody. An iterator counts the number of conjunct and disjunct movements and generates a recommendation if the ratio of steps to leaps is less than 3:1. If an interval is found to be greater than an octave, an additional recommendations are generated.

- Intervals
  - When melodies do contain leaps, these leaps should be of consonant rather than dissonant intervals. Consonant leaps include major or minor thirds or sixths, octaves, perfect fifths, and, in the case of more modern music, perfect fourths. Following a leap, the melody should move in stepwise motion in the opposite direction as the leap. Dissonant leaps include major or minor sevenths and tritones.

```
> interval.Interval(note1, note2.isConsonant()
> interval.Interval(noteStart=note1, noteEnd=note2).name
> interval.Interval(noteStart=note1, noteEnd=note2).semiSimpleName
```

  - The first of these functions will flag particular intervals as dissonant and then these will be saved to a list. These particular intervals are then identified and displayed to the user using the next two functions.

- Resolution
  - It is typical when finishing a melody to end with a note from the triad built off the first scale degree (usually either the root or the fifth). It is also possible to end with a note from the triad built off the fifth scale degree. The first option will provide a more complete sound than the second option. The second option will sound as if it is meant to go on.

```
> scale.MajorScale(root)
> scale.MinorScale(root)
```

– Using these two methods, it is possible to generate the pitches in a particular scale. The pitch in the 0, 1, 2, 4, or 6 position in the list are an acceptable ending to the melody. The preferred pitches would be 0 or 4 for a final ending and 1, 4, or 6 for an ending that would lead to something else.

Melody Feedback

Based on the areas of melodic development that are listed above, the tool will generate feedback to the user about what they could improve. It is possible to disregard all of the feedback and proceed with finalizing the melody. The feedback is there as a suggestion for those that may not know how to compose a melody or someone who is struggling to write a melody that they are pleased with how it sounds. It is not required to use any of the feedback.

That being said, once the feedback is presented, the user will be able to continue to make changes and reanalyze their melody. The feedback will reflect whatever changes they make. It is also the case that a user may choose to follow certain suggestions and not others. This is totally acceptable as the feedback is meant to help, but not to hinder creativity.

If the user is presented with feedback in terms of range, the tool will point out what exactly out of range and suggest that the user alter these notes or move them down an octave. If the user is given feedback about contour, the tool will show which specific leaps and following notes need addressed. If there is an issue with the ratio of conjunct to disjunct movements it will highlight the leaps and make suggestions about how to alter them.

If the user receives feedback related to the intervals within their melody, the tool will address the specific intervals that violate the given rules and use the transposition feature, that was highlighted earlier, to present alternative pitch options to correct the error. Additionally following each leap, the system will check to see if there is stepwise resolution using the same functions as the contour checking system. If this is not present, the system will recommend that the user add it and show them how. If the user gets feedback on their resolution of the melody, the system will recommend them alternate notes based on the key they have chosen.

Melody Review

Once the feedback has been given to the user, they have the option to either edit their melody or finalize and export it. If they choose to edit their melody, they can go back to any point in the creation process and change whatever they desire. Additionally, the feedback system can be rerun at any point to generate new feedback based on the changes.

Output

Once a melody has been created, the user has the option to export the MIDI file that was generated for playback. This file can be opened with any program or software that is compatible with MIDI as the file that this program generates is a standard MIDI file.

3.2 User Interface Development

The next phase of this project was to develop the user interface. Being that this is where other computer composers are lacking, the interface must be carefully planned and designed.

3.2.1 Methodology

This section details the background thinking and ideas that went into creating the user interface.

Design

It is a common debate in interface design (and design in general) whether form should follow function or vice versa. In the case of this project, it is necessary that the design of a particular element should reveal it's function. The user should be able to look at a particular button or slider, and without much effort, be able to figure out what it does.

Due the varying levels of music or computer science knowledge elements of the UI needed to be designed in a way that is simple and clear. There cannot be huge numbers of controls and there also cannot be any assumptions as to what a particular thing should do. The design of a component needs to spell out what it does.

Functions

When thinking about the functions of the interface, it must provide access to the initial setup parameters, display the note input system, provide instructions for use, provide access to the feedback system, and display the notation and playback window. Each of these functions are required in order to be able to fully interact with the composer.

Additionally, the access to these functions must be direct. The user must specifically be able to call each of these functions and see their output. There are other internal functions of the system, but these are not made to be accessible through the interface.

Layout

Due to the detail required in the note input system, the layout of the UI has been optimized for desktop browser. While it is possible to view and use the site on mobile, it is not recommended. The design of the note input system, in its current form, cannot be made small enough to be usable on a mobile device.

To make the tool easier to use, everything must be large enough and have adequate space in between. This is to prevent misclicks and errors when entering notes and accessing other functions.

3.2.2 Process

This section details the tools and processes that were used in the creation of the user interface.

Design

Functions

Layout

Chapter 4

Experimental Results

To evaluate the success of the composer, it seemed most appropriate to have a user experience study. It would not be a fair test of how effective the tool is at helping a person compose if the only people that were to test the tool have high level music and computer science knowledge. To truly understand how the tool performs, it needed to be placed in the hands of a general audience.

When designing software, two of the key elements are the user interface (UI) and user experience (UX). The UX is how the user feels about and responds to a product. It is based on how the tool is presented, acquired, and how it functions. This chapter of the document details the design of the UX study, its evaluation, and any potential threats to its validity.

4.1 User Experience Study

As a part of this study, there are several important metrics that had to be gauged in order to determine the effectiveness of the tool. These include:

- User Background
- System Availability
- System Performance
- System Accessibility
- User Impressions

Several questions were presented to the user about specific areas within each of these larger categories. This data was then used to determine the performance in the category as a whole. The specific categories, subcategories, and related questions are listed below.

4.1.1 User Background

This sections asks the user some demographic questions related to the user's previous experience and gets important connecting information.

Music Background

Rate your amount of music theory knowledge.

- No Knowledge
- Beginner
- Intermediate
- Advanced
- Expert

Rate your ability to read music.

- No Knowledge
- Beginner
- Intermediate
- Advanced
- Expert

Rate your knowledge related to instrumental and/or vocal performance.

- No Knowledge
- Beginner
- Intermediate
- Advanced
- Expert

Have you had musical instruction?

- Yes
- No

If yes, please select all of the following types that apply.

- Instrumental Instruction
- Vocal Instruction
- Theory Instruction
- History Instruction
- Composition Instruction

Have you written/composed music in the past?

- Yes
- No

Computer Science Background

Rate your amount of computer science knowledge.

- No Knowledge
- Beginner
- Intermediate
- Advanced
- Expert

Have you had some form of computer science instruction?

- Yes
- No

If yes, please select all of the following types that apply.

- Self Taught
- Online Instruction
- Classroom Instruction
- Professional Instruction

Composition Analysis

Would you like to submit your compositions for analysis?

- Yes
- No

If yes, would you like to be listed as the composer?

- Yes
- No

If yes, please provide how you wish to be named.

4.1.2 System Availability

This particular metric deals with whether or not the user was able to access the site and on what sort of device and conditions they had when the attempted to access the site.

General Access

Were you able to access and use the site?

- Yes
- No

Did you try on more than one occasion?

- Yes
- No

Platform

On what sort of device did you try to use the site?

- Desktop Computer
- Laptop
- Tablet
- Mobile Phone
- Other

Operating System

What operating system do you have on your device?

- Windows
- macOS
- iOS
- Linux
- Android
- Other

Browser

What web browser were you using?

- Chrome
- Safari
- Internet Explorer
- Microsoft Edge
- Firefox
- Other

4.1.3 System Performance

This metric deals with how quickly and completely the system responded to the user.

Load Time

Did each of the pages load quickly?

- Yes
- No

If no, with which page(s) did the issue occur?

Analysis Speed

Were the analysis, notation, and playback returned quickly?

- Yes
- No

Rendering Issues

Was there any part of the page that was missing?

- Yes
- No

If yes, with which page(s) did the issue occur?

4.1.4 System Accessibility

This metric deals with the elements of the UI.

Layout

Does the layout of the pages make sense to you?

- Yes
- No

If no, with which element(s) did you have an issue?

Design

Does the overall design of the page and the elements make sense to you?

- Yes
- No

If no, which which item(s) did you have an issue?

Sizing

Was everything on the page sized appropriately to make it easy to use?

- Yes
- No

If no, which which element(s) did you have an issue?

4.1.5 User Impressions

This sections determines the user's progress though the tool and their overall comments about using it.

Completion

Were you able to go through the entire composition process?

- Yes
- No

If no, where did you stop?
Why did you stop?

Frustrations

Was there any part of using this tool that was frustrating or upsetting?

- Yes
- No

If yes, with what part(s) of the tool did you experience this with?

Changes

Are there any changes that you would make to this tool?

- Yes
- No

If yes, what are they?

Comments

What other comments do you have for the developer of this tool?

4.2 Study Evaluation

4.3 Threats to Validity

Chapter 5

Discussion and Future Work

The computer composer that was designed, implemented, and tested as part of this project is very different that the other composers that currently exist. While their focus is on the composition algorithm and ability of the computer system itself, this tool attempted to bring the basic functions of a computer composition system to all kinds of users through the creation of a user interface with built in feedback mechanisms.

While this particular tool does not feature an advanced AI powered composition, it does contain an a user interface designed to be accessible for all users. This cannot be found on any of the other computer composers that are available. It is in this way that this tool sets itself apart.

There is most certainly some merit to the idea of designing a tool around the idea of accessibility. A person might build the most advanced tool ever created, but if it never reaches the hands of the average person, can it really have the same impact? While it is the case that not every technology is meant to shared with everyone, technologies that are used to create and make art should be available to everyone.

Many pieces of software within the music industry are priced in such a way that they are available only to the elite and industry professionals. This limits the possible creative output of these tools all for the sake of profit. By making these tools accessible to more people, the potential output from these tools would be dramatically increased.

This would lead to more music and more diverse music being developed and created. And while this tool is not a full fledged notation program, it exists to bring music composition to people from all sorts of educational backgrounds and levels of experience.

5.1 Summary of Results

This section details the successes and failures of each of the major areas of the project. This includes the development and functions of the composer, the design and deployment of the user interface, and the results and effectiveness of the user experience study.

5.1.1 Composer Development

Overall, the functions that were implemented within the composer are successful and perform their desired functions. Initially, the composer was meant to have more functionalities related to harmonization and writing parts for additional voices, but due to time constraints and developmental difficulties, the composer now only focuses on melody generation and revision.

While it is certainly possible to implement a harmonization function, this is far more complex than the creation of a singular voiced melody. There are many variables involved in harmonization and lots of room for user decisions and choices. This makes development of this sort of function more challenging and makes the system of recommendations to the user much more complex. The system is then much more difficult to use and there was simply not enough time to resolve all of theses issues.

However, the melody generation system is very robust and contains all of the functions that it needs in order to be able to help the user generate a melody. It may be much simpler without the harmonization system, but overall this helps the tool to better reach the overall goal of the project by making this something that anyone can use and understand.

5.1.2 User Interface Development

5.1.3 User Experience Study

5.2 Future Work

This section discusses the possible extensions and revisions that could be performed on the system. Some of these extensions were elements that were originally meant to be included in the system, but were eliminated for various reasons. It is quite possible, however, to include them with more time and resources.

5.2.1 Composer Extension

The most significant extensions to the composer would be the addition of functions for harmonization, additional voices, and supporting tracks.

Harmonization

The idea behind this function is to generate a chord progression and then voice these chords based on the notes in the melody. This would need to take into account voice leading and the movement between the melody and the bass note. Adding this function would allow the user to create full fledged songs rather than just melodies.

Additional Voices

This functions would use the chord progression generated by the harmonization function to write a complementary line to the melody. It would add another layer to the composition and create musical interest with possible dissonance and resolution.

Supporting Tracks

The function to create supporting tracks would generate some sort of percussion and/or bass instrument accompaniment to give the composition a more finalized sound. This adds yet another layer of musical interest and provides a metronomic backing to the playback.

5.2.2 User Interface Revisions

5.3 Conclusion

Bibliography

[1] Colombo, F., Brea, J., and Gerstner, W. Learning to Generate Music with BachProp. *CoRR abs/1812.06669* (2018).

[2] Colombo, F., and Gerstner, W. BachProp: Learning to Compose Music in Multiple Styles. *CoRR abs/1802.05162* (2018).

[3] Cuthbert, M. S. music21 Documentation. `https://web.mit.edu/music21/doc/index.html`, February 2020. Accessed: 2020-01-25.

[4] Encyclopedia Britannica. Computer music. `https://www.britannica.com/art/electronic-music/Computer-music`, February 2018. Accessed: 2019-12-01.

[5] Fernandez, J., and Vico, F. AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research 48* (Nov 2013), 513–582.

[6] Hadjeres, G., and Pachet, F. DeepBach: a Steerable Model for Bach chorales generation. *CoRR abs/1612.01010* (2016).

[7] Hiller, L., and Isaacson, L. *Machine Models of Music*. MIT Press, Cambridge, MA, USA, 1992.

[8] Jarre, J.-M. EON. `https://jeanmicheljarre.com`, 2019. Accessed: 2020-04-01.

[9] LilyPond Community. Manuals for LilyPond 2.20.0. `http://lilypond.org/website/manuals.html`, March 2020. Accessed: 2020-01-30.

[10] MIDIjs Team. MIDIjs – API Description. `https://www.midijs.net/midijs_api.html`. Accessed: 2020-03-18.

[11] Papadopoulos, A., Roy, P., and Pachet, F. Assisted Lead Sheet Composition Using FlowComposer. *Principles and Practice of Constraint Programming* (2016), 769–785.

[12] Savage, M. Jean-Michel Jarre launches 'infinite album'. `https://www.bbc.com/news/entertainment-arts-50335897`, November 2019. Accessed: 2020-03-30.

[13] Sony Computer Science Laboratories, Inc. Flow Machines – AI music–making. `https://www.flow-machines.com`, 2018. Accessed: 2019-11-10.

[14] Spaans, B. mingus 0.5.1 documentation. `https://bspaans.github.io/python-mingus/index.html#mingus`, 2015. Accessed: 2020-02-02.

[15] Teymuri, A. PythonInMusic – Python Wiki. `https://wiki.python.org/moin/PythonInMusic`, 2019. Accessed: 2019-10-01.