# PIANOTE: A SIGHT-READING PROGRAM THAT ALGORITHMICALLY GENERATES MUSIC BASED ON HUMAN PERFORMANCE

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Drew Schulz

June 2016

COMMITTEE MEMBERSHIP

| | |
|---|---|
| TITLE: | PiaNote: A Sight-Reading Program that Algorithmically Generates Music Based on Human Performance |
| AUTHOR: | Drew Schulz |
| DATE SUBMITTED: | June 2016 |
| COMMITTEE CHAIR: | John Clements, Ph.D.<br>Associate Professor of Computer Science |
| COMMITTEE MEMBER: | India D'Avignon, Ph.D.<br>Associate Professor of Music |
| COMMITTEE MEMBER: | Alexander Dekhtyar, Ph.D.<br>Professor of Computer Science |

ABSTRACT

PiaNote: A Sight-Reading Program that Algorithmically Generates Music Based on Human Performance

Drew Schulz

Sight-reading is the act of performing a piece of music at first sight. This can be a difficult task to master, because it requires extensive knowledge of music theory, practice, quick thinking, and most importantly, a wide variety of musical material. A musician can only effectively sight-read with a new piece of music. This not only requires many resources, but also musical pieces that are challenging while also within a player's abilities.

This thesis presents PiaNote, a sight-reading web application for pianists that algorithmically generates music based on human performance. PiaNote's goal is to alleviate some of the hassles pianists face when sight-reading. PiaNote presents musicians with algorithmically generated pieces, ensuring that a musician never sees the same piece of music twice. PiaNote also monitors player performances in order to intelligently present music that is challenging, but within the player's abilities. As a result, PiaNote offers a sight-reading experience that is tailored to the player.

On a broader level, this thesis explores different methods in effectively creating a sight-reading application. We evaluate PiaNote with a user study involving novice piano players. The players actively practice with PiaNote over three fifteen-minute sessions. At the end of the study, users are asked to determine whether PiaNote is an effective practice tool that improves both their confidence in sight-reading and their sight-reading abilities. Results suggest that PiaNote does improve user's sight-reading confidence and abilities, but further research must be conducted to clearly validate PiaNote's effectiveness. We conclude that PiaNote has potential to become an effective sight-reading application with slight improvements and further research.

# ACKNOWLEDGMENTS

Thanks to:

- Students in MU 163, Spring 2016, for taking the time to use PiaNote. Your participation was extremely helpful and valuable.
- Professor John Clements, for being an excellent advisor who constantly encouraged and challenged me. Your support has made this project possible.
- Professor India D'Avignon, for providing helpful information about sight-reading difficulties, and offering design ideas for PiaNote. Your generosity and ideas both inspired me and motivated me to make PiaNote what it is today.
- Benjamin Reveley, for taking hours out of your day to configure the piano lab for our user study.

TABLE OF CONTENTS

# LIST OF TABLES

LIST OF FIGURES

# Chapter 1

# INTRODUCTION

Mastering piano takes years of dedication and practice. Pianists use various methods to improve their reading skills. One such method is sight-reading, the act of performing a piece of music at first sight [15]. Sight-reading is taught frequently in musical education and is also considered to be "an important part of good musicianship" [16]. It forces one to understand musical notation and immediately apply that knowledge towards a performance.

## 1.1 Sight-Reading Difficulties

While sight-reading is important, it is difficult to master. A key component of the definition of sight-reading is "first sight". In order to effectively sight-read, one must always perform a new piece of music. This requires a substantial amount of resources—such as sight-reading books, piano playbooks, or instructor provided material—and can be quite costly for novice musicians. To make matters worse, sight-reading is also only effective if the difficulty of the musical piece is appropriate. If the piece of music is too easy, the musician may not be challenged, and therefore not improve his or her skill level. Likewise, if the piece is too difficult, the musician will only be frustrated by the piece and possibly lose motivation to sight-read.

## 1.2 Sight-Reading Resources

Various resources have been created in attempts to alleviate these sight-reading hassles. One example is the "Sight Reading" book series by James Bastien, which presents small exercises that increase in difficulty over time [1]. Other tools include

software such as Sight Reading Factory, a website that creates small sight-reading exercises for musicians, and Presto Keys, a desktop application that monitors a user's ability to identify random notes on a staff [10] [17]. While these tools present a wide variety of musical material, they do not assess a musician's proficiency level in order to present appropriate musical material. Sight Reading Factory and Presto Keys allow users to change difficulty levels and other settings, but constant setting adjustments may distract musicians from the main goal, sight-reading.

### 1.3 Our Contribution

This thesis presents PiaNote, a sight-reading application that presents musical exercises based on a user's performance and proficiency level. In contrast to sight-reading tools such as Sight Reading Factory and Presto Keys, PiaNote attempts to understand user difficulties and construct exercises that are appropriately challenging. The result is a user-tailored application that aims to improve pianists' sight-reading confidence and abilities.

PiaNote runs as a web application on Google Chrome. To interact with PiaNote, users perform musical pieces on a connected MIDI keyboard. PiaNote visually notifies users about their performance accuracy, and uses this information to present a new exercise.

To test the effectiveness of PiaNote, we conducted a small study on Cal Poly's MU 163 class. Participants were asked to sight-read PiaNote's exercises for three 15 minute sessions. Because the study was small, we are not able to draw clear conclusions about PiaNote. However the gathered data suggests that PiaNote is a useful tool with the potential to improve a musician's sight-reading confidence and abilities.

We discuss Background of Music Theory, MIDI, and Algorithmic Music Composition in Chapter 2, followed by Related Works in Chapter 3. Chapter 4 describes the Design and Application Overview of PiaNote, while Chapter 5 provides an in depth explanation of the Implementation. Chapter 6 presents the Validation of PiaNote. Lastly, Chapter 7 and 8 present Future Work on PiaNote and Concluding Remarks.

# Chapter 2

# BACKGROUND

This chapter provides information about topics that are relevant in future sections of this thesis. In short, we discuss music theory, pianos and MIDI keyboards, and algorithmic composition history.

## 2.1 Music Theory

Music is a grouping of sounds "that form a unity so as to convey a message, to communicate, or to entertain" [4]. More technically, music is an organized combination of pitch, rhythm, and harmony. Music is not only heard, but also written and read.

### 2.1.1 Musical Terms and Explanations

There are many musical terms presented in future sections. This section aims to define some of these terms for better understanding. All definitions can be found in the Oxford Dictionary of Music [15].

#### 2.1.1.1 Pitch and Intervals

Pitch is "the location of a sound in the tonal scale"[15]. In other words, pitch represents the frequency of a sound wave. Another word for pitch in music is **note**.

Pitch can be presented and interpreted in various ways. Western Music employs the notion of **pitch classes** in order to categorize various pitches. Western Music contains twelve different pitch classes, presented in Figure 2.1. Pitch classes are represented with the letters **A** through **G** in combination with the symbols ♯ and ♭,

| C | C♯/D♭ | D | D♯/E♭ | E | F | F♯/G♭ | G | G♯/A♭ | A | A♯/B♭ | B |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Figure 2.1: Western pitch classes. Notes are presented in ascending order with a semitone interval.**

called 'sharp' and 'flat' respectively. We loop through pitch classes in order to classify all pitches. For example, in ascending order, a **B** is always followed by a **C**, and a **G** is always followed by a **A**. While pitch classes technically begin with **A**, Western Music uses **C** as the base for all pitch classes, for historical reasons not discussed in this thesis.

The tonal distance between pitch classes is known as an **interval**. The term **interval** can have multiple meanings in music. When applied to pitch classes, there are different types of intervals: **semitones** and **whole-tones**. Semitones represent the tonal distance between two successive pitch classes [15], while a whole-tone represents the tonal distance equivalent to two consecutive semitones. For example, C-C♯ is a semitone, while C-D is a whole-tone. Twelve consecutive semitones are known as an **octave**.

#### 2.1.1.2 Melody and Scales

According to the Oxford Dictionary of Music, **melody** is "a succession of notes varying in pitch which have an organized and recognizable shape" [15]. Most musical pieces are based on **scales**. Scales are "a series of notes progressing up or down stepwise" [15]. For example, C-D-E-F-G-A-B-C is an example of an ascending scale.

It is possible to also classify different types of scales based on their intervals. Two types of scales are **major** scales and **natural minor** scales. Table 2.1 presents both a C major scale, and a C natural minor scale. We see that a major scale is always a sequence of whole-tone, whole-tone, semitone, whole-tone, whole-tone, whole-tone,

**Table 2.1: Table of scales that displays the structure of major and natural minor scales in both semitone notation and the key of C. Whole-tones and semitones are represented as *wt* and *st*, respectively.**

| Interval | Major Scale | | Natural Minor Scale | |
|---|---|---|---|---|
| | Semitone Notation | Example with C Base | Semitone Notation | Example with C Base |
| 1 | **base note** | C | **base note** | C |
| 2 | wt | D | wt | D |
| 3 | wt | E | st | E♭ |
| 4 | st | F | wt | F |
| 5 | wt | G | wt | G |
| 6 | wt | A | st | A♭ |
| 7 | wt | B | wt | B♭ |
| 8 | st | C | wt | C |

and semitone, regardless of the root. There is a major and natural minor scale for every pitch class.

The musical term **interval** can also be used to describe tonal distances in a scale. For example, C-E is a major 3rd interval, because **E** is the 3rd note in the C major scale. Likewise, C-E♭ is a minor 3rd interval, because **E**♭ is the third note in the C natural minor scale.

On a broader level, we can classify major and minor intervals with semitones. A major third consists of four semitones, while a minor third consists of three semitones. For example, C-E is always a major third, while E-G is always a minor third.

**Table 2.2: This table displays common musical durations.**

| Term | Duration |
|---|---|
| Whole Note | 2 Half Notes |
| Half Note | 2 Quarter Notes |
| Quarter Note | $\frac{1}{2}$ of a Half Note |
| Eighth Note | $\frac{1}{2}$ of a Quarter Note |
| Sixteenth Note | $\frac{1}{2}$ of an Eighth Note |
| 32nd Note | $\frac{1}{2}$ of a Sixteenth Note |

#### 2.1.1.3 Key

In classical music, the **key** of a musical piece represents the scale that the piece is built upon. In general, we say that a piece of music is in the key of C major if the melody contains only pitch classes from the C major scale. There are 24 different keys that are represented by all major and natural minor scales.

#### 2.1.1.4 Rhythm

According to the Oxford Dictionary of Music, rhythm encapsulates all components in music pertaining to time [15]. A **rhythm** can be described as a forward moving group of durations based on a specified beat. A **beat** is a unit of time, often measured in beats per minute (bpm). **Duration** is the length of time a note should be held.

In western music, there are various terms used to classify common rhythms, which we present in Table 2.2. All of these terms are described relative to one another. The actual duration of a rhythm depends upon a **tempo**, which represents the speed of a musical piece. A tempo assigns one rhythm as the beat, and specifies its duration. For example, if a tempo assigns a quarter note to 120 bpm, then the duration of a

quarter note is 0.5 seconds. With this information, we can calculate the appropriate durations for all other rhythms in the piece.

#### 2.1.1.5 Harmony and Polyphony

Harmony is "the simultaneous sounding of notes" [15]. Another word for harmony is **polyphony**. Harmony is created when multiple instruments play different notes at the same moment, or when a single instrument with polyphonic abilities, such as a piano, plays two or more notes at once. Examples of harmonies include **chords**, which are a "combination of notes, but usually not fewer than [three]" [15]. Other examples of harmonies are two notes that are separated by a certain interval. For example, the harmony of C-E is a third, as well as E-G. Likewise, a **C** played with a **G** is a major fifth.

### 2.1.2 Musical Notation

There are various ways to visualize music. This thesis focuses on piano specifically, so this section discusses piano sheet music notation.

#### 2.1.2.1 Staves and Notes

Piano notation consists of two **staves**. A staff is a series of five equidistant parallel lines where notes can be placed [15]. Together, both staves are called a **grand staff**. Each line and space corresponds to one of the pitch classes A - G. To represent sharps and flats, ♯ or ♭ is placed before the note.

Both staves have **clefs**, which are symbols identifying the location of notes. The grand staff contains a treble clef and a bass clef. The treble clef identifies **G** as the second line from the bottom of the staff. The bass clef marks **F** as the second line from the top of the staff. We can derive the locations of all other notes from this

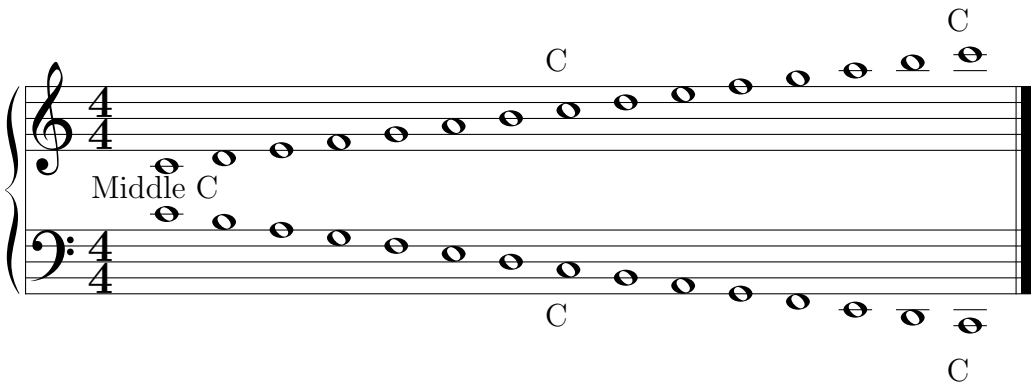



**Figure 2.2: The Piano Grand Staff which includes a treble clef and a bass clef. The staves connect at Middle C.**

**Table 2.3: Symbols representing each rhythm down to a 32nd note.**

| Name | Note Representation | Rest Representation |
|---|---|---|
| Whole | 𝅝 | 𝄻 |
| Half | 𝅗𝅥 | 𝄼 |
| Quarter | ♩ | 𝄽 |
| Eighth | ♪ | 𝄾 |
| Sixteenth | 𝅘𝅥𝅯 | 𝄿 |
| 32nd | 𝅘𝅥𝅰 | 𝅀 |

information. An example of the grand staff can be seen in Figure 2.2. Both staves meet at a **C**, which is known as "Middle C".

#### 2.1.2.2 Rhythm Notation

Notes and rests are drawn in a certain shape to represent note rhythms on a staff. Table 2.3 displays each symbol and the rhythm it represents.

To represent periods of time, staves are broken up into standard sections called **measures**. Measures contain the number of beats specified in a **time signature**. A time signature is a fraction placed at the beginning of a staff. The denominator represents the "unit of measurement in relation to the whole-note", while the numer-

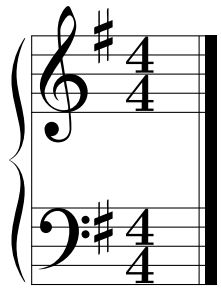


**Figure 2.3: Example Grand Staff in $\frac{4}{4}$ time in G major.**

ator represents the "number of units in each measure" [15]. For example, the time signature $\frac{4}{4}$ states that a measure represents a duration equivalent to four quarter notes. Figure 2.3 shows an example of $\frac{4}{4}$ time.

#### 2.1.2.3 Key Signature

A **key signature** displays sharps and flats that exist in a piece's key. Key signatures place ♯ or ♭ symbols at the front of the staff on lines or spaces corresponding to sharp or flat notes. For example, in the key of G major, the corresponding scale is G-A-B-C-D-E-F♯-G. Because there is one sharp in this key, **F**♯, a sharp is place on the corresponding line or space that represents **F**. Figure 2.3 shows an example of a grand staff in the key of G major.

## 2.2 Piano and MIDI Keyboards

### 2.2.1 Standard Piano

A piano typically has 88 keys, with 52 white keys and 36 black keys. White keys represent notes without any sharps or flats, while black keys represent notes with sharps or flats. Hitting a key activates wooden hammers that strike a string in order

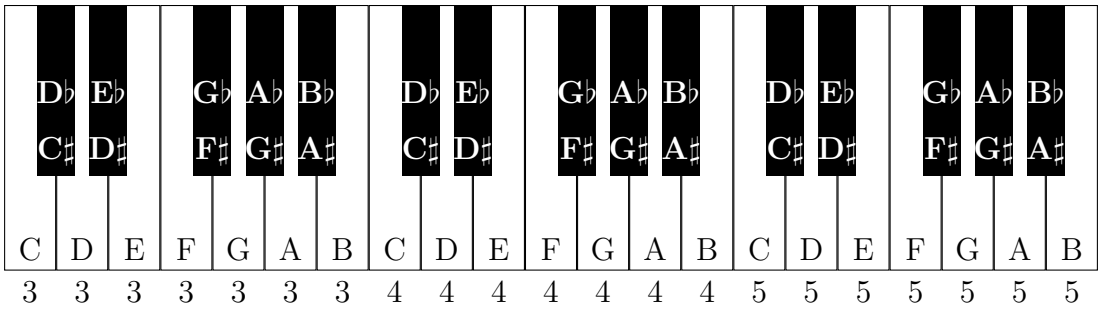



**Figure 2.4: Key mappings for piano.**

to produce a pitch. Keys can be played together to form harmonies, and keys can be hit in succession to form melodies.

In the context of piano keys, pitches increase from left to right. A standard piano consists of seven octaves. With so many octaves, it is confusing to locate the specific note **C**, because seven of them exist. To alleviate this, each key on a piano is specified by a tone and a number. For example, middle C is "C4". A standard piano spans from A0 to C8. An image of key mappings can be found in Figure 2.4.

### 2.2.2 MIDI Keyboards

MIDI keyboards are devices that are arranged like a piano, with both white and black keys. Hitting a key on a MIDI keyboard transmits a MIDI signal through a USB connection. MIDI stands for Musical Instrument Digital Interface [9]. Each MIDI signal is represented by messages such as "note-on" and "note-off". Within these messages is information about the note played. Notes are represented by numbers ranging from 0 to 127. Middle C is represented by MIDI note 60, for example. MIDI makes it easy for computers to understand musical input because it is compact and organized. PiaNote uses a MIDI connection to gather input from users.

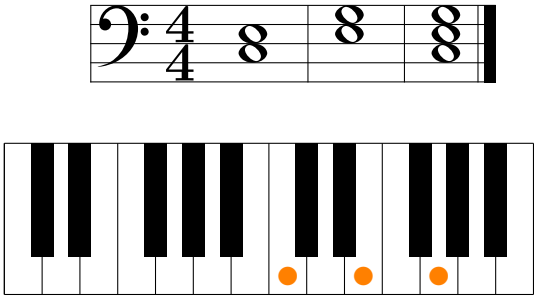


**Figure 2.5: The construction of a major triad chord. The first harmony is a major third (C-E), and the second harmony is a minor third (E-G). Combined, they create the C major triad (C-E-G). The piano displays the placement of a C major triad.**

### 2.2.3 Chords

Pianists are able to perform many different types of chords. As mentioned in Section 2.1.1, chords are harmonies that contain three or more notes. There are a plethora of different types of chords. In this section, we briefly discuss chords most relevant to this thesis.

#### 2.2.3.1 Triad

A triad is a three note chord that is a grouping of two successive thirds [15]. There are many different types of triads. We will go over just major and minor triads. A major triad is composed of a lower major third and and upper minor third. A minor triad is composed of a lower minor third and an upper major third. Figure 2.5 shows an example of a major triad chord.

#### 2.2.3.2 Six-Four Chord

A six-four chord consists of a "bass note with a 6th and 4th above it"[15]. Six-four chords are really just second inversion triads. An **inversion** is a rearrangement of

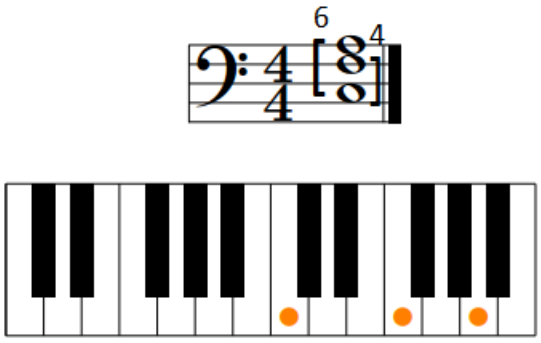


**Figure 2.6: A Six-Four chord with a C base. This chord contains all of the same notes as an F major triad. The piano displays the placement of the six four chord.**

notes in a chord [15]. For example, the chord E-G-C is the first inversion of the chord C-E-G, because the base of the chord is the third, not the root. A six-four chord is a second inversion triad, where the fifth is the base of the chord. Using our previous example, the second inversion of C-E-G is G-C-E. An example of another six-four chord, C-F-A, is shown in Figure 2.6.

#### 2.2.3.3 Dominant Seventh Chord

A dominant seventh chord is a dominant chord with an added 7th. A **dominant chord** is a triad with its root on the fifth note of a scale. For example, in the key of C, the dominant chord is the G major triad, G-B-D. The dominant seventh chord adds a seventh to the dominant, forming G-B-D-F in our last example. Like triads, dominant seventh chords can be inverted. Sometimes, dominant sevenths also drop the fifth of the chord so only the root, third, and seventh remain. Figure 2.7 displays two forms of the dominant seventh chord G-B-D-F. The first is a full dominant seventh without any inversions. The second entry is a first inversion of the dominant seventh, as the base is **B**, with an omission of the fifth, **D**. The resulting chord is B-F-G. This form is common in piano when preceded or followed by a C major triad, because hand movement is minimal.

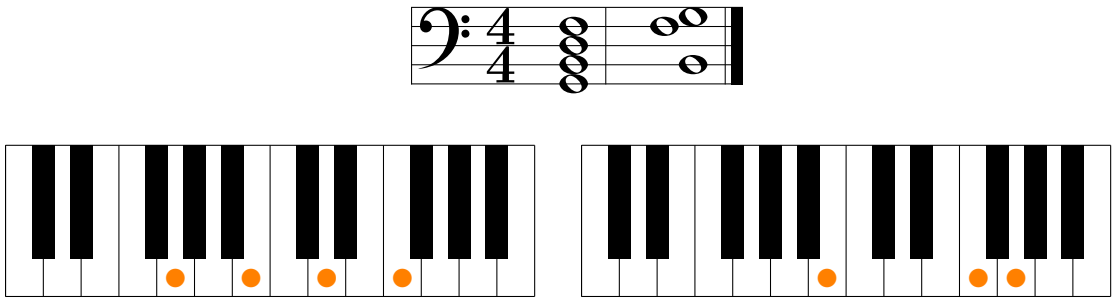


**Figure 2.7: Two forms of a dominant seventh chord in the key of C. The left chord is in root position, and consists of G-B-D-F, while the right consists of B-F-G. The chord on the right is an inversion of the first chord. The piano images display both versions of the chord.**

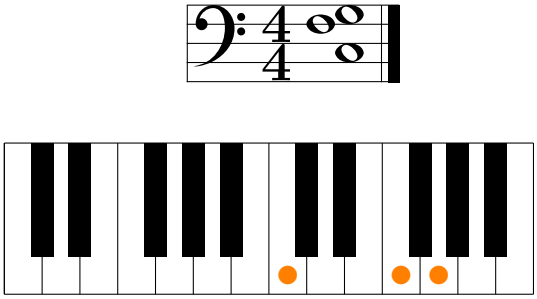


**Figure 2.8: A C suspended chord. Usually this chord will be resolved to a C major triad. The piano displays the placement of a C suspended chord.**

#### 2.2.3.4 Suspended Chord

A suspended chord is a triad where the third is replaced by either a perfect fourth or a major second. These chords are usually followed by a major triad. For example, the suspended chord C-F-G usually resolves to a C-E-G. Figure 2.8 shows example notation of a suspended chord.

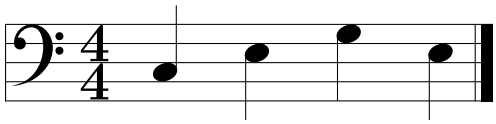


**Figure 2.9: A C major chord as a broken chord. This forms the melody C,E,G,E.**

#### 2.2.3.5 Broken Chord

A broken chord, or **arpeggio**, is a chord with its notes played in succession, instead of simultaneously [15]. Broken chords are often found in left hand melodies on piano. Figure 2.9 shows an example of a broken chord.

## 2.3 ABC Notation

ABC notation is musical notation used to typeset tunes that are stored in an ASCII format. It was developed by Chris Walshaw in 1993 and was created primarily to transcribe Irish folktunes, but has been expanded to a wider variety of music [25] [24].

ABC notation makes it easy to create pieces in an easy to read format. Many software libraries have been developed to interpret ABC notation, such as the JavaScript library abcjs [20]. ABC's structure makes it not only easy to parse, but also easy to construct programmatically.

### 2.3.1 ABC Notation Basics

All ABC notation is plain ASCII text. There are many different aspects to the ABC notation language. The basics are explained in the context of the example ABC shown in Figure 2.10.

```
X: 1 %Id number, if there are multiple songs
T: Song Title %song title
M: 3/4 %time signature
L: 1/16 %the default note length
K: G %key signature
G4 A4 B4 | A4 [GB]4 A2B2 | G8 G'4| F'2E'2 D'8 ||
```

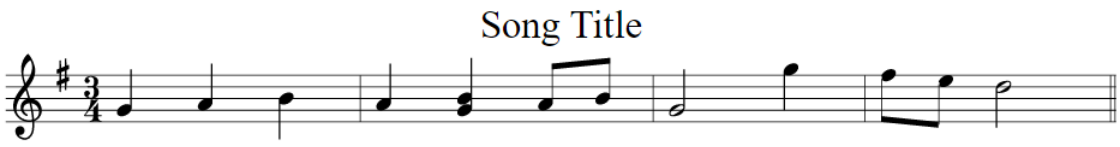



**Figure 2.10: ABC notation example. This figure shows both the text representation and the render of ABC notation.**

#### 2.3.1.1 Piece Configuration

The musical piece can be configured in many ways. One can assign a title to the piece or set other components like the time signature. As can be seen in Figure 2.10, The title is set to "Song Title", and the time signature, represented by "M" for meter, is set to $\frac{3}{4}$.

#### 2.3.1.2 Constructing a Piece

The melody of a piece is written as a succession of letters representing pitches, and also the pipe symbol, |, to denote a bar line. Pitches are represented by both the letter and modifier characters, such as " , " for an octave below, and " ' " for an octave above. The duration of a note is set by adding a number to the end of the note. For example, in Figure 2.10, the first note, "G4" is a **G** that is 4 times the length of the default note length, $\frac{1}{16}$. In other words, the note is a quarter note. The default note length can be changed in the piece configuration.

Chords can also be created by wrapping notes in brackets, such as the "GB" harmony in the second measure of Figure 2.10. Beamed notes, often seen in durations like eighth and sixteenth notes, are automatic in ABC notation as long as the desired notes are placed together without a space. This can be seen in the end of the second measure of Figure 2.10.

## 2.4 Algorithmic Music Composition

An algorithm is defined as a set of steps carried out to achieve a result. Algorithmic composition is the act of creating a melody through a series of steps and rules. There has been much research in the field of algorithmic composition, including stochastic methods like Markov chains as well as genetic algorithms.

### 2.4.1 Stochastic Music Composition

A stochastic composition algorithm uses probabilities and random actions. In essence, it chooses random notes and rhythms to generate a melody. One common form of stochastic algorithms are Markov chains. Markov chains are systems "in which the present outcome depends on a number of previous outcomes" [14]. Transitions from each state is weighted, so while Markov Chains are random, each outcome is not independent and also probabilistic. An example of a Markov Chain can be seen in Figure 2.11.

David Cope's Experiments in Musical Intelligence (EMI) uses Markov chains to develop melodies. EMI attempts to generate a melody based on the style of a collection of musical pieces [5].

In order to use a Markov chain, a chain of states, transitions, and weights must be constructed. In order to make a musically interesting Markov model, chains are

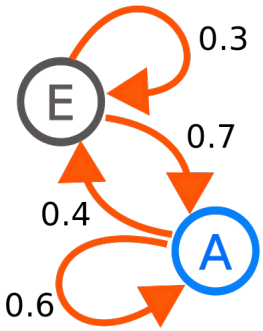



**Figure 2.11: An example of a Markov chain. The numbers on the edges represent weights, or the probability of transitioning from one state to another [26].**

often constructed by an existing piece of music [14]. Because of this, it is difficult to use Markov models without any previous musical input.

### 2.4.2 Genetic Algorithms

A genetic algorithm is an artificial intelligence technique in which a population of possible solutions, generated at random, undergoes natural selection according to their effectiveness, indicated by a fitness function [14]. This fitness function determines whether a possible solution is allowed to "breed". All fit solutions create new generations of solutions through a method called crossover, where two individuals swap part of their bits with each other, creating new individuals. Another operation called mutation also occurs, where random bits of the new individuals are flipped, in order to "mutate" their traits a bit. This entire process repeats until there is a desirable set of solutions that can be used.

Genetic algorithms have been used frequently in algorithmic music composition. One example is Al Biles' GenJam, which improvises jazz solos and reacts to human performances using genetic algorithms [2].

### 2.4.3 Controversy

Algorithmic composition has suffered from significant resistance, with individuals claiming that computers cannot truly create music when it is composed from previous pieces [5]. Resistance also comes from the notion that computers should not replace humans in the art of creating music [8]. The work presented in this thesis does not aim to compete with humans when creating music. Instead, its goal is to aid both new and experienced pianists in practicing sight-reading.

# Chapter 3

# RELATED WORK

Many sight-reading tools have been created to aid pianists, and dynamic difficulty adjustment has been researched for games and educational exercises. This chapter explores both areas that relate to PiaNote.

## 3.1 Sight-Reading Software

Numerous software tools have been created to help musicians learn piano as well as sight-read. This section details musical education tools similar to PiaNote.

### 3.1.1 The Piano Tutor

The Piano Tutor, developed by Dannenberg et al. at Carnegie Mellon University, is a software tool that assesses a musician's ability to play piano in order to present new piano lessons [6]. This tool was developed in 1993 and is a multimedia intelligent tutoring system. It consists of an electric piano, as well as a sheet music user interface to display lessons. The Piano Tutor has a preset database of musical lessons with meta-data associated to them. This meta-data is used to select appropriate lessons for beginner piano students based on their performance and progress. The Piano Tutor assumes that an initial user has no musical knowledge on start up, so it walks users through basic musical concepts initially. The Piano Tutor and PiaNote are very similar, but PiaNote does not contain a finite database full of lessons. Instead, it algorithmically generates lessons, creating an even more tailored experience for users. PiaNote is also accessible from anywhere as a web application.

### 3.1.2 PianoFORTE

PianoFORTE was developed at National University of Singapore by Smolier et al. [22]. It goes farther than The Piano Tutor developed at Carnegie Mellon by monitoring user's stylistic performance through MIDI. Stylistic performance includes dynamics, tempo, articulation, and synchronization. PianoFORTE monitors a user performance and will visually display this information on top of the score. A piano instructor can view this graphical information to start a dialogue with the student about their performance and how it can be improved. PianoFORTE is meant to enhance the piano lesson experience through multimedia. PiaNote takes this a step further by monitoring a user's performance and using that information to generate new material for the user to practice.

### 3.1.3 Sight Reading Factory

Sight Reading Factory is a commercial online tool, created by GraceNotes LLC, dedicated to helping musicians sight-read [10]. It supports numerous instrumental notation such as piano, guitar, and percussion, as well as many key signatures, time signatures, and difficulty levels. After configuring all settings, Sight Reading Factory algorithmically generates a piece of sheet music for the user to play. New pieces of music can be generated by clicking a button on the page. PiaNote takes this a step further and not only presents a user with randomly generated sheet music, but also monitors the user's ability to play it. It then uses this information to generate new pieces of music to read. PiaNote also automatically adjusts the difficulty for a user based on his or her abilities and performances.

### 3.1.4 Presto Keys

Presto Keys is an open source desktop application, created by Marty Papamanolis, that aims to help users learn the piano without any musical experience, and also aid existing pianists in practicing [17]. The software can be used with a mouse or a MIDI keyboard, and will randomly generate notes for a user to play. The software scrolls through the randomly generated notes and monitors a user's performance in regards to correct notes and durations. If a user misses a note, Presto Keys notifies the user visually and provides feedback on the correct note name and position on the piano. Presto Keys has various options and features, as well as a difficulty level selection. PiaNote enhances the experience Presto Keys offers by monitoring a user's playing in order to actively focus in one area of music and provide new music in that area. PiaNote also automatically selects the difficulty for users.

## 3.2 Dynamic Difficulty Adjustment

There has been a lot of research done in the area of dynamic difficulty adjustment, especially in game development. We touch on a couple examples that are most relevant to PiaNote.

### 3.2.1 Triage Training System

Tayama et al. present a triage training system that automatically adjusts difficulty based on a trainee's performance on different exercises [23]. The system has three components that contribute to the difficulty of an exercise: the number of people to triage, treating patients with sudden changes in symptoms, and transferring patients to the hospital. Based on the performance in a certain component, its difficulty is adjusted for the next exercise. The difficulty is scored with the following heuristic:

80% accuracy and above increases the component difficulty, 60% - 80% keeps the component at the current difficulty, and 60% and below decreases the component difficulty. All of the components are scored independently and adjusted independently. The difficulty of the current exercise is determined only by the previous performance and not by a collection of past performances.

Tayama et al. conducts a study to test their system by presenting a pre-test to users, then allowing them to use the tool freely, and concluding with a post-test to test their progress. Results show that dynamic difficulty adjustment improved results on the post-test.

PiaNote also uses a component difficulty adjustment that is very similar to the triage training system. However, PiaNote's adjustments are done according to musical components.

### 3.2.2 Temporal Data Driven DDA

Zook et al. present a dynamic difficulty model for games that uses challenge tailoring [28]. Challenge tailoring "is the problem of matching the difficulty of skill-based events over the course of a game to a specific player's abilities"[28]. Zook et al. focus on player performance instead of difficulty level. The argument is that player performance is easier to measure and is objective, while difficulty is subjective. The focus of challenge tailoring in this paper is in adventure role-playing games. An important factor in this game that is used to measure performance is the use of magical spells.

Tensors are created in order to form a player performance model. A tensor is essentially a three-dimensional matrix. Each cube inside the tensor resembles a component of the first, second, and third axis. The axes in the player performance tensor are players, spell types, and the time of the performance monitoring. This model

allows the player's performance to be tracked over time. Tensor factorization is used by means of CP decomposition to weight each of the components, such as spell types and users. This data is used to produce new scenarios for players.

Zook et al. conducted a study of 32 participants to test the effectiveness of the tensor model. Players were asked to play through 11 battles and report their perceived difficulty in the battle, as well as their enjoyment. The results of the study suggest that the tensor model is able to make useful predictions in future game scenarios with small amounts of training data.

# Chapter 4

# APPLICATION OVERVIEW

PiaNote strives to aid novice pianists in sight-reading practice. The application automatically adjusts difficulty levels based on human performances, and algorithmically creates pieces of music that it perceives will be most helpful to the user. This chapter details the broad design and usability of PiaNote. Chapter 5 discusses deeper implementation of the adaption algorithm and exercise creation.

## 4.1 Application Design

PiaNote is designed to keep users focused on one main task, sight-reading. Because of this, PiaNote's interface is very simple and streamlined.

### 4.1.1 Application Flow

The application flow is very simple. After log-in, users are presented with a piece. Users are asked to perform the piece with a tempo of their choice. PiaNote has a built-in metronome that can be set between 80 bpm and 120 bpm.

After performing, users analyze their performance and select a new piece with the click of a button. A visual of the application flow is seen in Figure 4.1.

The cyclic nature of PiaNote creates an unending presentation of new music to sight-read. Users are able to play as many pieces as they please in one session, and can take a break at any time. The application saves user data after each piece is played, so a user is able to use the tool days later without losing progress.

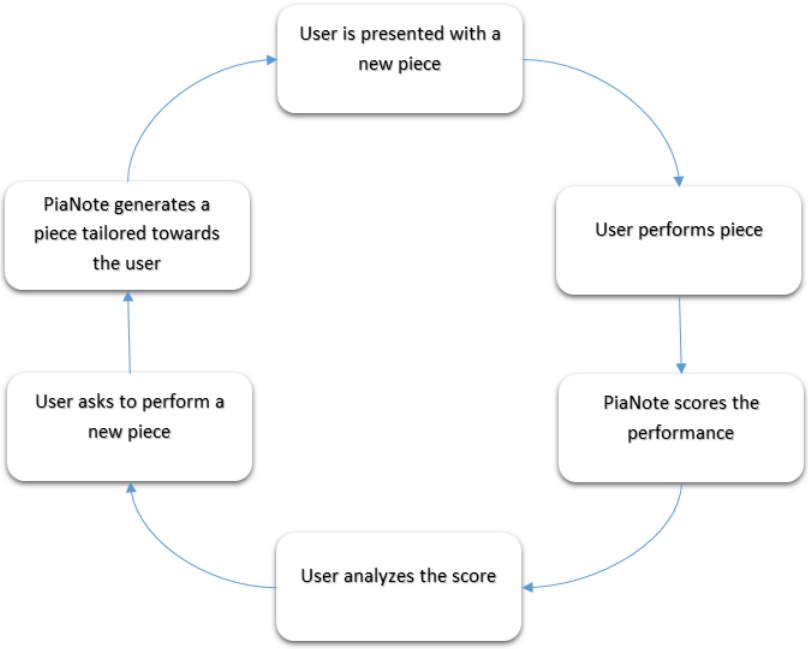



**Figure 4.1: PiaNote application flow.**

### 4.1.2 Application Usability

PiaNote is designed to be unburdensome for users. The application interface is very streamlined because users are interfacing with both a MIDI keyboard and a computer. The system is modeled to be entirely accessible through mouse clicks, with the exception of the log-in screen. The application is easy to use with a touch screen or a computer screen with stylus support. Because the piano lab on Cal Poly's campus contains touch screen computers, this application interface is ideal. Users can interact with the application without having to switch back and forth between a computer keyboard and a MIDI keyboard.

### 4.1.3 Application Presentation

PiaNote is a simple web application with a log-in page and a main application page. The main application page is presented in Figure 4.2. This page contains a four measure piece of sheet music for the user to read. Depending on the type of piece, there may be a subtitle describing its focus, as seen in Figure 4.3. The user presses a record button to begin recording his or her performance.

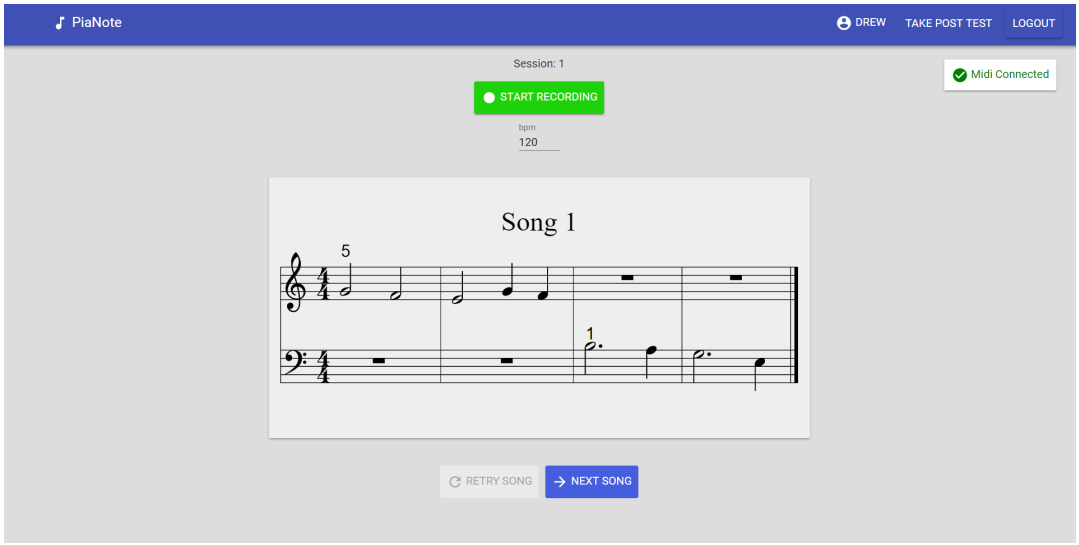



**Figure 4.2: PiaNote main application page.**

#### 4.1.3.1 Performance Grading

Once a user has performed, the application evaluates the player's accuracy. An overall performance score is not presented to the user after finishing a piece to avoid unnecessary frustration when a user continually performs poorly. For example, it may frustrate a struggling user to constantly see that his or her performance accuracy is 30%. Sight-reading is already a stressful activity that requires ample amounts of practice to improve, so instead of presenting users with a pass/fail metric, we present a metric that focuses on the musical components users need work on. The aim is to

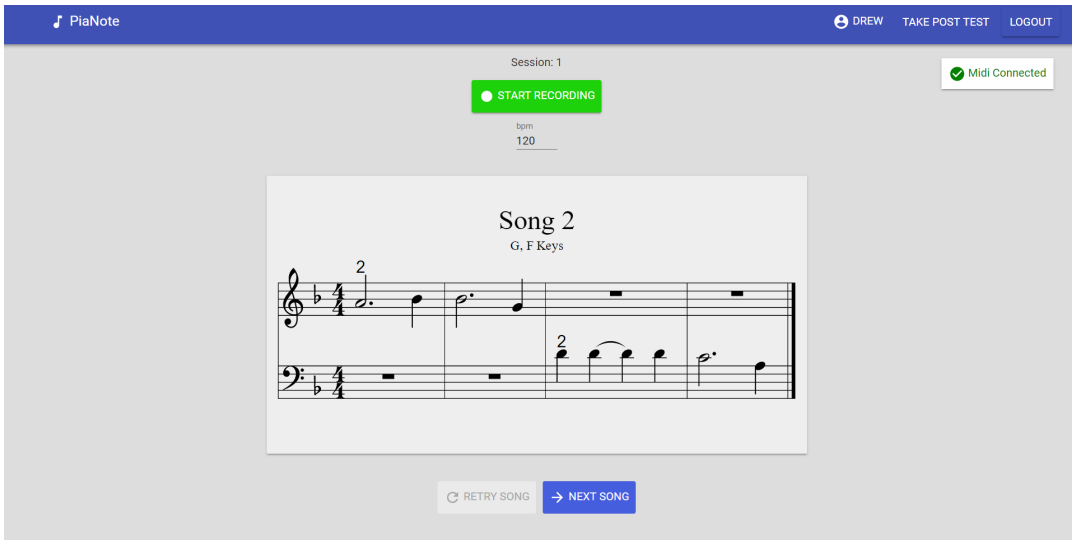



**Figure 4.3: PiaNote piece with a subtitle to denote its focus. In this case, the piece focuses on pieces in the key of G or F.**

encourage users to focus more on sight-reading practice rather than level or lesson completion.

Instead of presenting users with a numerical score, PiaNote highlights notes on the staff based on a user's performance. We choose colors as a means for scoring because they are easy to discern and clearly identify how users perform in certain areas. An example of the note highlights can be seen in Figure 4.4. PiaNote presents a scoring key to describe the meaning of the colors. If a note is green, the user played both the note and rhythm correctly, while a red note represents an incorrect note and rhythm. In cases where the user plays the correct pitch, but a wrong rhythm, the note is highlighted in blue. Likewise, when the user plays a wrong pitch but a correct rhythm, the note is highlighted in purple. In Figure 4.4, the second to last note has an incorrect pitch, because it is purple.

To retrieve further information about the performed piece, users are able to listen to the expected piece by clicking the "Play Expected Piece" button. Likewise, they are also able to hear their own performance by clicking the "Play Your Performance"

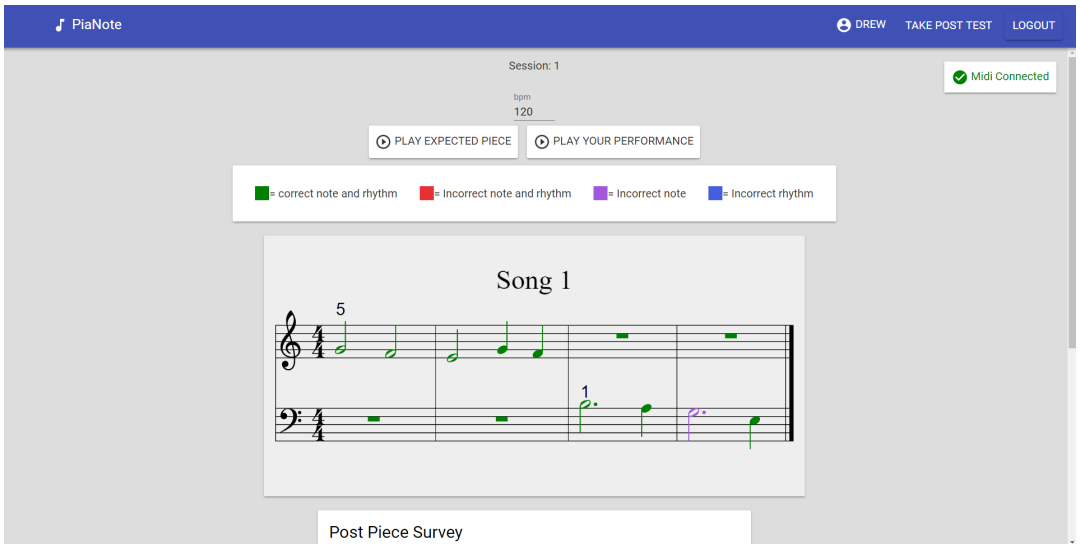



**Figure 4.4: PiaNote player scoring. Notes in the piece are highlighted based on a user's performance. Users also have the option to hear the expected piece and their performance.**

button. This is included so that users can get auditory feedback about how the piece was supposed to sound.

Lastly, users can dissect their performance even more by clicking on each individual note in the piece. This will launch a dialog displaying the expected note and rhythm, as well as the performed note and rhythm. A visual of this dialog can be seen in Figure 4.5.

## 4.2 Technology Overview

Many technologies are used to create PiaNote in browser. This section details the software design model used, as well as the use of various software libraries.

### 4.2.1 Software Model

PiaNote is a simple web application, with most of the application logic done on the client. The server is merely used for database storage. Because of this, most of the

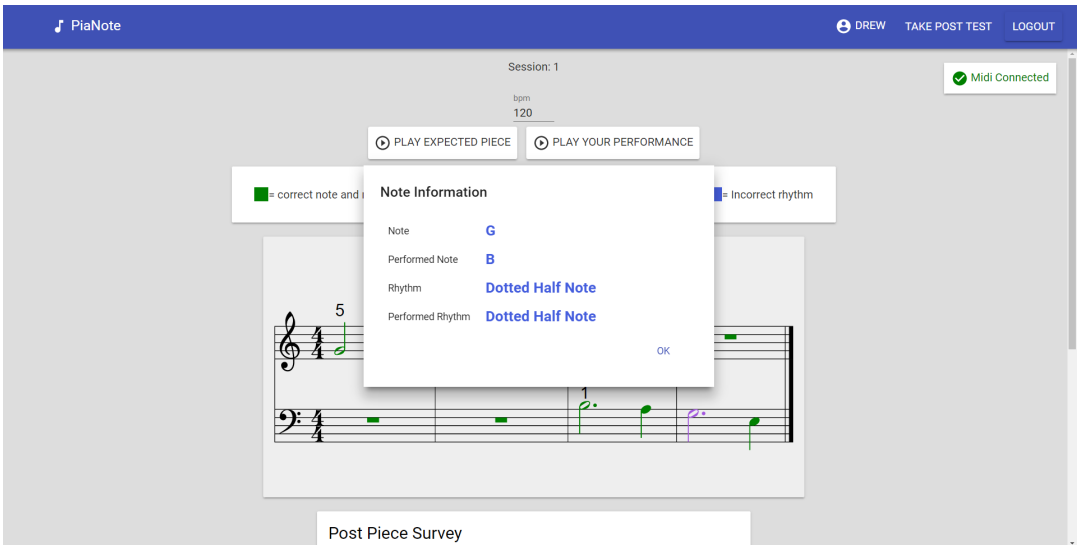



**Figure 4.5: PiaNote note dialogs display more information about a performed note. This dialog displays the missed G note in the last measure.**

application code, such as sheet music creation, user models and analytics, and MIDI capturing is written in JavaScript. The server is a means of data dumping, because this application is a prototype built for a user study. Not too much focus is given on web application design, but instead on effective difficulty adjustments for sight-reading applications as a whole. With this in mind, the software model is shown in Figure 4.6.

#### 4.2.1.1 Server Implementation

PiaNote's server is implemented with Flask, a web server microframework in Python[11]. Flask makes it easy to quickly create a simple web server. PiaNote's server has six endpoints: register, login, logout, home, load, and score. Load will attempt to restore a user's data if they are not logging in for the first time. Score is the only POST call to the server, which sends a snapshot of the user model and the latest musical performance.

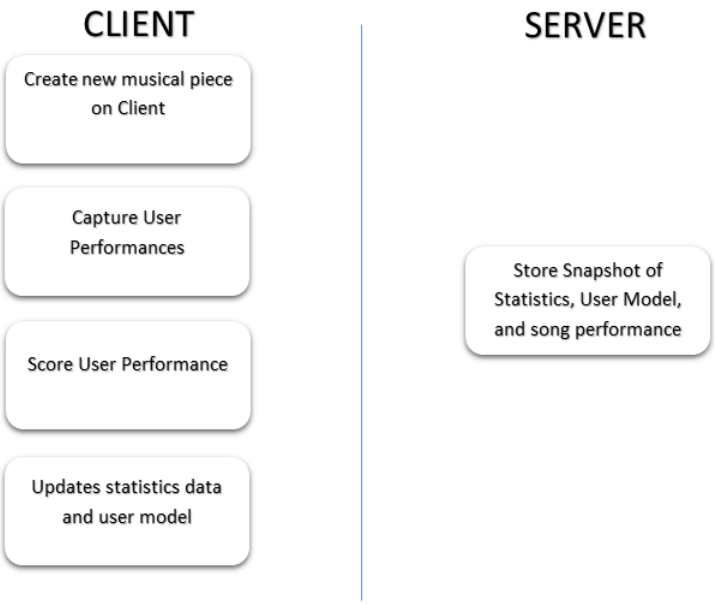



**Figure 4.6: PiaNote Software Model detailing the jobs executed on the client and server.**

### 4.2.2 Musical Components

PiaNote contains various musical components such as musical notation and musical playback. We describe technologies to make these features possible in PiaNote.

#### 4.2.2.1 Musical Notation with abcjs

Abcjs is a JavaScript library that renders musical notation given in ABC notation [20]. ABC notation is discussed previously in Section 2.3. Abcjs not only offers the ability to translate plain string text into a sheet music render, but also has the ability to highlight components of the musical score.

#### 4.2.2.2 Musical Playback with MIDI.js

Performance playback is made possible with the help of MIDI.js, a JavaScript audio sequencing library [7]. MIDI.js contains a MIDI player that can play a MIDI stream

in a particular soundfont. MIDI.js contains a grand piano general MIDI soundfont, which is used for playback in PiaNote.

### 4.2.3 MIDI Connection

MIDI devices are able to connect to PiaNote by means of the Web MIDI API, a JavaScript API with functionality for connecting and monitoring MIDI hardware [27]. The Web MIDI API notices USB connection and pairs it to a web application by means of JavaScript callbacks. Likewise, the API has callbacks for MIDI messages such as "note on" and "note off". The Web MIDI API is currently only supported natively in Google Chrome [12]. Because of this, the application currently only works in Google Chrome.

# Chapter 5

# IMPLEMENTATION

PiaNote contributes to the development of sight-reading software with an algorithm to generate music based on a user's previous performances. This algorithm can be broken into three parts: monitoring a user's performance, tracking a user's progress with a user model, and creating new musical pieces. This chapter explains all three algorithms.

There is a wide variety of music that can be created and read. PiaNote offers a small sliver of musical material in order to control the pieces' musicality and difficulty level. This sliver of musical material is still large enough to create a wide variety of musical pieces. The pieces created by PiaNote are loosely modeled after pieces presented in *Sight Reading, Level 2*, by James Bastien [1].

In short, PiaNote focuses on five components of a musical piece:

- Key Signature
- Time Signature
- Song Type
- Intervals
- Rhythms

Song type refers to the overall structure of the piece, such as chord types or the use of the left and right hand. It does not refer to a certain mood or genre.

We discuss how the algorithm takes these components into account when presenting new songs to users.

## 5.1 Monitoring User Performance

This section outlines methods to fully monitor a user's performance, general hardships in monitoring musical performances, and PiaNote's solutions to these hardships.

### 5.1.1 Scoring a Performance

PiaNote scores a performance in five key areas: key signature, time signature, song type, intervals, and rhythms. An exercise in PiaNote is four measures. Users' performances are monitored through MIDI and matched to the exercise once they have finished the piece.

MIDI data is able to capture the key played, when a key is pressed, and when it is released. From this information, we can determine the pitch of the key and its duration to measure **note accuracy**, **rhythm accuracy**, **note and rhythm accuracy**, and **interval accuracy**. Calculations for these accuracies are shown below.

$$RhythmAccuracy = \frac{RhythmsHit}{TotalRhythms} \tag{5.1}$$

$$NoteAccuracy = \frac{NotesHit}{TotalNotes} \tag{5.2}$$

$$NoteRhythmAccuracy = \frac{NotesHit+RhythmsHit}{TotalNotes+TotalRhythms} \tag{5.3}$$

$$IntervalAccuracy = \frac{IntervalsHit}{TotalIntervals} \tag{5.4}$$

Notice that we only measure rhythm accuracy as a hit or miss. For simplicity, we do not measure the severity of a rhythm miss. With these accuracies, we can gener-

ate a score for five different musical areas: key signature, time signature, intervals, rhythms, and song type.

Rhythm score is computed directly from rhythm accuracy. Components related to piece structure, such as song type and time signature, are scored with note-rhythm accuracy. Key signature is scored with the note accuracy, and intervals are scored by the interval accuracy.

#### 5.1.1.1 Explanation and Limitations of Scoring System

Many components are scored broadly. For example, key signature score models all note accuracy in an exercise instead of only the accuracy of notes with sharps and flats. Note that some scores are generated from similar criteria, such as song type score and time signature score.

PiaNote's exercises are very short, so it is difficult to create distinct methods when scoring individual components. If scoring is done on a very specific metric, this may result in very harsh scores if the user happens to struggle. For example, let's say that key signature score only measured accuracies of notes with sharps and flats. If a user performs a piece in the key of G, but there is only one **F**♯ in the piece, the user's key signature score would either be 100% or 0%. Because these accuracies are used in order to adjust difficulty, this score seems extremely harsh, especially if the user hits all other pitches in the piece. As another example, let's say the user is presented with a piece in the key of C, which has no sharps or flats. There is no effective way to measure key signature scores because C has no sharps or flats.

In order to avoid these edge cases, key signature scoring is based upon a user's ability to play within a certain key. As a result, the key signature score is calculated from overall note accuracy, making key signature scores less complex.

Time signature scores are difficult to calculate because a time signature only affects the duration of the measure. In PiaNote, chords change every measure. In different time signatures, a user's left hand will switch chords at a different frequency, affecting duration. Different time signatures also introduce a different set of rhythms. For example, a piece in $\frac{3}{4}$ time will never present a whole note, because its duration is longer than that of a measure. Because time signatures heavily alter the structure of the piece, we use note-rhythm accuracy to measure its performances.

While the scoring system may be broad, it still allows us to monitor a user's proficiency in various aspects of a performance.

### 5.1.2 Performance Monitoring Difficulties

When playing music, especially piano, it is possible to easily miss a note or accidentally hit an unexpected note. With a strict monitoring system, these actions may not be addressed, which can lead to an incorrect performance evaluation.

#### 5.1.2.1 Missing a Note

In the case of missing a note or a beat, we look at Figure 5.1. In this figure, we see that the musician has forgotten the first note entirely, but has hit all successive notes correctly. A strict monitoring system may not be aware that a user missed only one note and may interpret the performance as a complete failure. This will happen if the monitoring system treats the performance as a sequence of notes to match with the expected piece. In the example presented in Figure 5.1, the monitoring system would produce the following pairs: E-G, G-A, A-G, G-C. A simple mistake on the player's part can skew their results on an entire performance if the monitoring system is too strict.

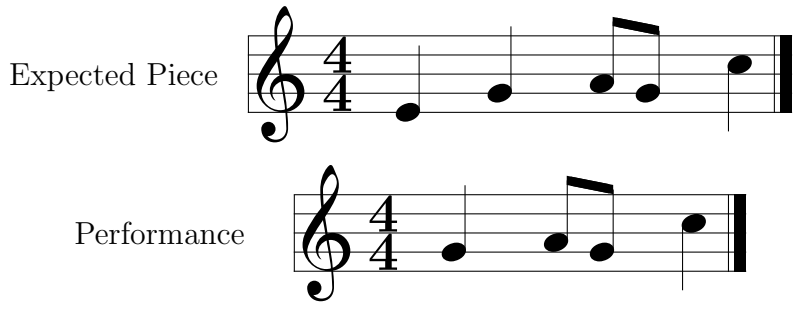



**Figure 5.1: A performed piece where the user has missed the first note.**

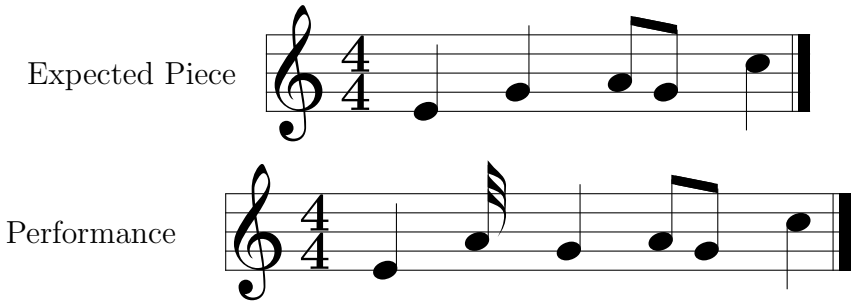



**Figure 5.2: A performed piece where the user's finger slipped between the first and second note.**

#### 5.1.2.2 Playing an Unexpected Note

Strict monitoring problems can also occur when a user introduces a note that is not presented in the piece. This can happen frequently, as a player's finger may slip and hit a wrong note for a small amount of time. An example of this is shown in Figure 5.2. A strict monitoring system may not notice that noise data exists in the performance, and that removal of such noise would result in a better performance score.

A strict scoring system can be both misleading and also frustrating for users, as they will have to play a piece perfectly to get any useful feedback. This completely defeats the purpose of sight-reading, which is assessing the ability to play a piece of music correctly at first sight.

### 5.1.3 Performance Monitoring Discussion

Our goal is to match a user's performance to an expected piece with the best accuracy. One option is to look at a performance as a set of points in two dimensional space—pitch and start time. If we treat both the expected piece and the performance as sets of points in this space, it may seem intuitive to match points using a nearest neighbor approach. All performance points would be matched to the nearest expected point. However, this approach treats all notes on an individual basis, and notes are not regarded in the context of an entire performance. A performance is always a forward moving sequence. Our gathered performance data is not just a cloud of events, but an ordered succession of events. It may be more valuable to use this to our advantage in order to create a more accurate match, instead of the nearest neighbor approach.

### 5.1.4 PiaNote's Solution to Strict Monitoring

In an attempt to remove noise data in performances and also correct for missed notes, PiaNote uses an edit-distance alignment algorithm to match the player's performance with the expected piece.

Edit-distance is a common measure used to determine the similarity of two strings [19]. Edit-distance is usually comprised of a score of insertions, deletions, and substitutions. For example, given two strings, "day" and "bay", the edit distance is 1, as one substitution must be made to make "day" into "bay". Similarly, the edit distance between "balloon" and "saloon" is 2, because the "b" is switched with an "s", and an "l" must be deleted—known as a deletion—from "balloon" in order to become "saloon".

```
B A L L O O N
| | | | | | |
S A L _ O O N
```

**Figure 5.3: Edit-distance alignment of two strings, "balloon" and "saloon".**

The edit-distance algorithm can be extended to align both strings. For example, "balloon" and "saloon" would be aligned as shown in Figure 5.3. We compute edit-distance and align two strings with a dynamic programming algorithm.

Edit-distance alignment has been used in DNA sequencing algorithms, such as Smith-Waterman's local alignment algorithm [21]. PiaNote uses a similar algorithm in an attempt to best align a player's performance with the expected piece. A brief description of the algorithm is mentioned below:

1. Separate the performance and expected piece into left-hand and right-hand melodies.
2. Flatten melodies into a sequence of pitch-rhythm pairs.
3. Compute the edit-distance between the performance and expected piece sequences.
4. Align the performance and expected piece sequences according to the edit-distance.

#### 5.1.4.1 Separating Melodies

First, we need to split both the performance and expected piece into left-hand and right-hand melodies. This is trivial for our expected piece, because both melodies are clearly separated in their own staff. For a performance, we have to match raw data to a hand position manually with a pitch split point.

The split point is determined by the key of the expected piece. For example, all pieces in the key of C have a pitch split point at middle C, or MIDI number 60. All pitches below the split point are assigned to the left-hand melody, while all pitches above the split point are assigned to the right-hand melody. On expected piece creation, we ensure that all left hand melodies remain below the piece's split point, and all right hand melodies remain above the split point.

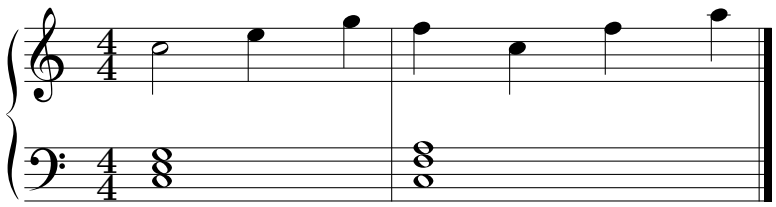
#### 5.1.4.2 Flattening Melodies

Once we match the melodies to a certain hand, we flatten the melodies into a sequence of pitch-rhythm pairs. The sequence is sorted by the note's start time, and then its pitch. Sorting by pitches allows us to flatten harmonies such as chords. We display a flattened expected piece in Figure 5.4.

#### 5.1.4.3 Calculating Edit-Distance

After flattening the melodies, we calculate the edit-distance between the expected sequences and the performance sequences. Our sequence elements have two components, in contrast to strings which have one component. As a result, we have five edit-distance measures instead of the original three measures of insertion, deletion, and substitution. The five measures are:

- Insertion
- Deletion
- Rhythm Substitution
- Pitch Substitution
- Rhythm and Pitch Substitution



```
//note: {p: <MIDI pitch>, r: <duration>}

leftHand = [{p: 48, r: 4}, {p: 52, r: 4}, {p: 55, r: 4},
            {p: 48, r: 4}, {p: 53, r: 4}, {p: 57, r: 4}]

rightHand = [{p: 72, r: 2}, {p: 76, r: 1}, {p: 79, r: 1},
             {p: 77, r: 1}, {p: 72, r: 1}, {p: 77, r: 1},
             {p: 81, r: 1}]
```

**Figure 5.4: A musical piece flattened into two sequences of pitch-rhythm pairs.**

**Table 5.1: Match scores for edit distance calculations. The score for individual component correctness and incorrectness is 3 and -3, respectively.**

| Scenario | Score |
|---|---|
| Correct Note and Rhythm | 6 |
| Correct Note and Incorrect Rhythm | 0 |
| Incorrect Note and Correct Rhythm | 0 |
| Incorrect Note and Rhythm | -6 |
| Insertion | -9 |
| Deletion | -9 |

An insertion resembles a musician's tendency to hit extra notes that are not presented in the piece. Likewise, a deletion represents a player missing key notes in the piece. Looking at Figure 5.1, we can see that the edit distance between the two pieces is one deletion, as the first note **G** must be deleted from the expected piece in order to correctly match the performed piece. Likewise, in Figure 5.2, we can see that the edit-distance between the two pieces is one insertion, as the 32nd note **A** must be inserted into the expected piece in order for it to match the performance.

#### 5.1.4.4 Edit-Distance and Alignment

Because edit-distance measures have been tweaked to work with musical data, the alignment algorithm must also be tweaked in order to support the different edit-distance measures.

In order to correctly match a performed piece to an expected piece, scores are created for each scenario. The scores are listed in Table 5.1.

The goal of the alignment is to align the pieces of music that results in the highest score. A correct component earns a score of +3, while an incorrect component earns a score of -3. This is why the second and third rows in Table 5.1 have a score of

0, because those scenarios contain a correct component and an incorrect component, resulting in the score 3 - 3 = 0.

While the magnitude of these scores may not be important, the weight of the scores are very important. For example, insertion and deletion must be the worst score in this model. Added notes and missed beats are rare, and should only occur if the rest of the performed piece matches better with or without the note. If a player truly performs the wrong pitch and rhythm, we want a substitution to occur, not an insertion or deletion.

Calculating the optimal edit-distance between the expected piece and the performed piece allows small portions of the piece to be aligned correctly. As sight-reading can be hectic, players are bound to skip over notes, read only parts of the piece correctly, or hit certain keys unintentionally. PiaNote's goal is to score the player in the most accurate way possible in order to correctly notice the player's struggles.

### 5.1.5 Difficulty in Measuring Note Duration

It can be difficult to consistently measure the rhythm that a user intends to play because computers have such precise timing. For example, consider the piece displayed in Figure 5.5. This piece contains two notes, **C** and **E**. Both notes are quarter notes, but when played, they may not be held for the same amount of time. The musician will lift a finger off of **C** right before placing another finger on **E**. As a result, **C** may be held slightly less than **E**. This is even more evident when both notes are the same pitch, as the player must deliberately lift a finger early in order to play the same note again. When listening to music, this time difference may not be noticed, but the milliseconds of difference is still captured by a computer.

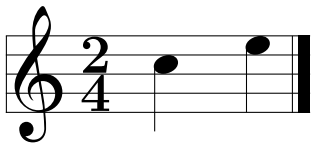


**Figure 5.5: Two quarter notes, C and E.**

Overprecise duration monitoring can cause problems and frustration among piano players. The player may intend to play a quarter note, but a strict system may interpret their duration as a dotted eighth note, for instance.

Interpreting duration can be especially difficult when there is no existing baseline to match with the raw data. This is experienced with sheet music programs such as Finale's PrintMusic Software [18]. PrintMusic is a sheet music application that contains a feature called hyperscribe. With hyperscribe, musicians can transcribe a performance from a MIDI device to rendered sheet music. Finale notes that when doing this, quantization is used to measure note duration. The default quantization in PrintMusic is an eighth note. Any rhythm detected by the computer that is less than an eighth note will round up to an eighth note when printed [18]. This is done to ensure that the sheet music is not cluttered with bizarre rhythms. However, in certain situations this still may not be the rhythm that the musician intended to play. PrintMusic's solution is to give users the ability to change quantification settings, and also manually edit the piece after using hyperscribe.

### 5.1.6 PiaNote's Solution to Measuring Duration

PiaNote uses a form of quantization when measuring duration that is similar to PrintMusic's solution [18]. Because PiaNote has an expected piece to match performance durations to, it becomes even easier to infer the rhythm that the player intended to play.

**Figure 5.6: Supported note lengths in PiaNote.**

When PiaNote compares durations, it measures the difference in the performance duration and the expected duration. If the duration is within a certain epsilon of expected duration, PiaNote concludes that the user has played the correct rhythm. Otherwise, PiaNote concludes that the performed rhythm is incorrect, and uses quantization to round the duration to the nearest supported rhythm. The quantization setting in PiaNote measures durations down to a sixteenth note. However, PiaNote only rounds to a specific set of note lengths. Supported note lengths are listed in Figure 5.6.

PiaNote only rounds to supported note lengths because introducing bizarre rhythms measured down to a sixteenth of a beat, such as a double dotted quarter note, may be unnecessary and possibly more confusing to users.

## 5.2 User Model

PiaNote contains a user model that tracks user performances and determines the components a user should focus on. There are two key factors in the user model: the user's difficulty level, and performance statistics.

### 5.2.1 User Difficulty Level

PiaNote focuses on five key areas of music: key signature, time signature, song type, intervals, and rhythms. We choose these five areas because we are able to create a difficulty ranking for each. The difficulty levels for each component are listed in Table 5.2.

**Table 5.2: Component difficulty levels. An X represents a nonexistent level since some components have less than five.**

| Component | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 |
|---|---|---|---|---|---|
| Key | No Sharps or Flats | 1 Sharp or Flat | 2 Sharps or Flats | 3 Sharps or Flats | X |
| Time | $\frac{4}{4}$ Time | $\frac{3}{4}$ Time | $\frac{2}{4}$ Time | X | X |
| Intervals | 1st, 2nds, 3rds | 5ths | 4ths | X | X |
| Rhythms | Whole, half, quarter | eighth | dotted quarter and eighth | sixteenth | dotted eighth and sixteenth |
| Song Type | Separate Hand | Simple Chords | Full Chords | Mixed Chords | Arpeggios |

Some component levels have natural difficulty progression. Key signature difficulty is based upon the number of sharps and flats. Rhythm difficulty progresses in the form of shorter rhythms and the introduction of shorter dotted rhythms.

Time signature difficulty progression is based on the frequency of chord changes. Chords change every measure, so $\frac{2}{4}$ represents the hardest difficulty.

Interval difficulty progresses generally based on the size of the interval jump. We rank 4ths higher than 5ths because 4ths aren't as easy to discern on a piece of sheet music. Looking at Figure 5.7, we see that G-D, a major fifth, has both notes placed on lines in the staff. In the second measure, the interval G-C, a 4th, contains one note on a line, and the other on a space. All 4th intervals have one note on a line and one on a space, making it harder for musicians to discern the extent of the jump. In contrast, it is clear that a fifth is two lines above or below the current note.

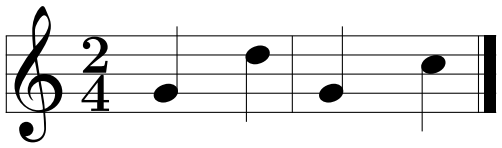


**Figure 5.7: A piece of music showing a 5th and a 4th interval. The first measure contains G-D, while the second contains G-C.**

The most interesting of these components is the song type. Song type affects the left-hand involvement in a piece of music. Descriptions of all song types can be found in Table 5.3.

**Table 5.3: Descriptions of all song types in PiaNote.**

| Song Type | Description |
|---|---|
| Separate Hand Piece | The right hand plays two measures, then the left hand plays two measures. |
| Simple Chord Piece | The right hand plays a melody, while the left hand plays two note harmonies. |
| Simple Full Chord Piece | The right hand plays a melody, while the left hand plays simple three note chords with minimal hand position changes. |
| Mixed Chord Piece | The right hand plays a melody, while the left hand plays chords with a possibility of hand position changes. |
| Arpeggio Piece | The right hand plays a melody, while the left hand plays arpeggios. |

A user's current level is an aggregation of the component levels. We represent this as a vector, $\langle$k, t, s, i, r$\rangle$, where **k** represents key signature, **t** represents time signature, **s** represents song type, **i** represents intervals, and **r** represents rhythms. For example, the level $\langle 1, 3, 2, 1, 2\rangle$ represents pieces in the key of C, 2/4 Time, with

<key, time, song type, intervals, rhythms>

| | |
|---|---|
| 1. <2, 1, 1, 1, 1> | 9. <2, 1, 1, 2, 1> |
| 2. <1, 2, 1, 1, 1> | 10. <1, 2, 1, 2, 1> |
| 3. <2, 2, 1, 1, 1> | 11. <2, 2, 1, 2, 1> |
| 4. <1, 1, 2, 1, 1> | 12. <1, 1, 2, 2, 1> |
| 5. <2, 1, 2, 1, 1> | 13. <2, 1, 2, 2, 1> |
| 6. <1, 2, 2, 1, 1> | 14. <1, 2, 2, 2, 1> |
| 7. <2, 2, 2, 1, 1> | 15. <2, 2, 2, 2, 1> |
| 8. <1, 1, 1, 2, 1> | |

**Figure 5.8: The first 15 levels of PiaNote. Components are listed in the following order: Key, Time, Song Type, Interval, Rhythm**

simple chords, 1st 2nd and 3rd intervals, and eighth note rhythms. A level is selected based on the PiaNote's dynamic difficulty algorithm.

### 5.2.2 PiaNote's Dynamic Difficulty Algorithm

PiaNote uses previous performance accuracies to assess whether a difficulty level should increase, decrease, or remain the same. Similar to the multi-component difficulty adjustment presented in the Triage Training System by Tayama et al., PiaNote increases and decreases component levels based on accuracies in the last performance [23]. PiaNote also includes a clear level progression so that lessons build upon each other and also focus on certain areas. We present PiaNote's first fifteen levels in Figure 5.8.

We argue that component difficulty adjustment without level progression can lead to lesson imbalance. Component difficulty adjustment only challenges users in areas where they are competent. For example, if one user has excellent rhythm perfor-

mance, but poor key signature performance, component adjustment would continually increase rhythm levels, and key signature levels would stay low. This user would get exposure to more rhythms, but not exposure to new key signatures. In other words, a user may never improve in certain areas.

We want to create balanced lessons that expose users to a wide variety of musical concepts, and also focus on areas where users struggle. We include a leveling system to limit lesson imbalance. Component difficulty cannot increase past the current level. For example, if a user has excellent rhythm performance and poor key signature performance in level $\langle 2, 1, 2, 1, 1 \rangle$, the rhythm difficulty will not increase. A user must perform well in all components in order to pass the level. This way, users get equal exposure to all components.

#### 5.2.2.1 Progressing Through Levels

We can think of levels in Figure 5.8 as **target levels** that users must pass in order to progress. We gradually introduce new concepts over time, creating harder musical exercises. For example, PiaNote will present a new type of key signature, a new type of time signature, and then present an exercise with both the new key signature and the new time signature.

Target levels contain **focus components**, which are the components with the highest difficulty. Looking at Figure 5.8, the focus components for target level 5 are key signature and song type. Likewise, the focus components for target level 15 are key signature, time signature, song type, and intervals. The point of a target level is to present exercises focusing on certain musical areas.

To pass a target level, a user must score above 80% on all components. If the user passes, they progress to the next level. If any component accuracies are less than 80%, the user does not pass the level.

If a user does not pass a level, PiaNote performs component difficulty adjustments similar to the Triage Training System in Tayama et al. [23]. If a component score is between 70% and 80%, the component difficulty does not change. If the component score is less than 70%, the component difficulty decreases, as long as it is not a focus component.

Because the purpose of a target level is to drill users on focus components, their difficulty level remains fixed, no matter the performance. If a focus component score is less than 70%, all other components decrease in difficulty. This is done to decrease the overall difficulty while still introducing new concepts.

If a component difficulty decreases from its target difficulty, it can return to the target if subsequent scores are greater than 80%. If a user continually struggles on a target level, PiaNote reverts to the previous target level.

Figure 5.9 displays pseudocode for PiaNote's dynamic difficulty adjustments. As can be observed, PiaNote's difficulty progression is linear if users are performing well, but it strays from the progression if users are struggling with certain musical components. The hope is that PiaNote is able to recognize broad user difficulties in order to present relevant music to sight-read.

To further visualize this algorithm, we present an example of a user at the current level $\langle 3,2,2,2,2 \rangle$. Table 5.4 displays a sample session for a user.

We see in Table 5.4 that PiaNote recognizes when the user begins to struggle. Looking at song 3, the user struggles with rhythms. Because rhythm is not a focus component for song 3, PiaNote lowers the difficulty of rhythm so that the user receives an easier piece for song 4.

In song 6, one of the components the user struggles with is song type, which is a focus component. Because of this, PiaNote lowers the difficulty of all other

```
if all component accuracies > 80%:
    increase to the next target level

else if user has struggled in the target level 3 times:
    decrease to the previous target level

else:
    if focus component accuracies < 70%:
        lower all other component levels by one

    else:
        for all components:
            if component score > 80%:
                if component difficulty
                  < target component difficulty:
                    raise component difficulty by one

            else if component score < 70%:
                decrease component difficulty by one
```

**Figure 5.9: Dynamic difficulty adjustment algorithm for PiaNote. Difficulty progression generally follows a linear path, but strays from linear progression if a user is struggling. The algorithm aims to bring users back to the current path once sufficient practice has been done in weak areas.**

**Table 5.4: An example user session. This table displays the level the user is presented with, and how their accuracies affect the difficulty presented to them. The level represents the component level, ordered in the following way: Key Signature, Time Signature, Song Type, Intervals, Rhythms.**

| Song Number | Level ($\langle$k,t,s,i,r$\rangle$) | Focus | Performance Accuracies (0-1) |
|---|---|---|---|
| 1 | $\langle$3,2,2,2,2$\rangle$ | {k} | $\langle$1.0, 1.0, 1.0, 1.0, 1.0$\rangle$ |
| 2 | $\langle$2,3,2,2,2$\rangle$ | {t} | $\langle$1.0, 1.0, 1.0, 1.0, 1.0$\rangle$ |
| 3 | $\langle$3,3,2,2,2$\rangle$ | {k t} | $\langle$0.9, 0.75, 0.75, 0.75, 0.6$\rangle$ |
| 4 | $\langle$3,3,2,2,1$\rangle$ | {k t} | $\langle$0.8, 0.8, 0.8, 0.8, 0.8$\rangle$ |
| 5 | $\langle$3,3,2,2,2$\rangle$ | {k t} | $\langle$0.8, 0.8, 0.8, 0.8, 0.8$\rangle$ |
| 6 | $\langle$2,2,3,2,2$\rangle$ | {s} | $\langle$0.8, 0.6, 0.6, 0.8, 0.4$\rangle$ |
| 7 | $\langle$1,1,3,1,1$\rangle$ | {s} | $\langle$0.8, 0.8, 0.8, 0.8, 0.8$\rangle$ |
| 8 | $\langle$2,2,3,2,2$\rangle$ | {s} | $\langle$0.7, 0.75, 0.75, 0.7, 0.8$\rangle$ |

components for song 7. Once the user performs competently in the song type, we see the difficulty increase back to the target, $\langle$2,2,3,2,2$\rangle$.

### 5.2.3 Performance Statistics

Performance statistics are kept for specific musical components. These statistics are very helpful when further tuning a musical piece. For instance, PiaNote will present pieces in either the key of G or F when the key signature is focused on level 2. Without performance statistics, it is difficult to know which key signature to present to the user. It is a possibility that a user performs very well in the key of G, but struggles in the key of F. In order to focus on student difficulties, PiaNote should focus on the key of F for this user instead of the key of G. This can only be done with the use of performance statistics.

Every user has a set of performance statistics that are updated each time a piece is performed. Performance statistics carry very specific information. There are four

types of statistics categories: key signature statistics, time signature statistics, song type statistics, and rhythm statistics.

Key signature statistics store the user's average accuracy for each key. For example, if a user performs a piece in the key of G, the key signature score is added to the collected accuracies for that key. Other component statistics store accuracies very similarly. Time signature statistics store accuracies for every time signature, rhythm statistics store accuracies for every individual rhythm, and song type statistics store accuracies for every song type. We explain how performance statistics affect exercise creation in Section 5.3.3.

## 5.3 Creating Musical Pieces

After monitoring and analyzing the performance to adjust difficulty, PiaNote constructs a new musical piece. All pieces are generated algorithmically. PiaNote uses a unique algorithm to generate random pieces in order to strictly control the structure and difficulty of the exercises. This section explains the process of musical piece creation.

### 5.3.1 Structure of a Musical Piece

PiaNote builds the structure of the musical piece before selecting notes and rhythms. Key signature, time signature, and chords are the three main construction components.

#### 5.3.1.1 Choosing Key Signature and Time Signature

PiaNote selects the key and time signatures based on the current difficulty level. For example, if the user's key signature difficulty is level 3, PiaNote picks a key signature

that contains up to 2 sharps or flats, which is the set {C, G, F, D, B♭}. If the key signature is a focus component, PiaNote picks a key signature by strictly adhering to the current difficulty. With our previous example, if key signature level is both 3 and a focus component, PiaNote chooses between {D, B♭}, the keys with 2 sharps or flats. Time signature choices are done in a similar manner.

#### 5.3.1.2 Choosing Chords

Chords are the last structural component in PiaNote's exercises. PiaNote assigns a chord to each measure in the piece. Each exercise contains four measures, so PiaNote generates four chords per lesson. The first and fourth chord are always the tonic chord. A tonic chord is a triad formed from the first note in the scale. For example, the C major triad is the tonic chord in the key of C. In PiaNote, a piece in the key of C will always contain a C major chord on the first and fourth measure, while a piece in B♭ will contain B♭ major chords on measures one and four. This is done to make exercises appear more musical. Starting and ending on the tonic chord is natural in music and appealing to the ear. PiaNote selects chords in measures two and three randomly based on the song type. Table 5.5 displays supported chords for various song types in the key of C.

**Table 5.5: This table displays the supported chords for each song type when in the key of C.**

| Song Type | Chords Allowed |
|---|---|
| Separate Hand Piece | {C, F, G, Dm, Am, Em} |
| Simple Chord Piece | {C, F, G} |
| Simple Full Chord Piece | {C, F, G} |
| Mixed Chord Piece | {C, F, G, Dm, Am, Em} |
| Arpeggio Chord Piece | {C, F, G, Dm, Am, Em} |

At this stage, choosing chords for each measure is only a means of assigning a certain tone to a measure. Chords such as triads and six four chords are not actually constructed in this step. Constructing such chords is done when PiaNote selects notes and rhythms.

### 5.3.2 Creating Notes and Rhythms

Once the structure of the piece is created, PiaNote creates notes and rhythms to complete the piece. PiaNote selects rhythms first, and assigns pitches to the rhythms afterwards.

#### 5.3.2.1 Constructing Rhythms

PiaNote chooses rhythms randomly according to the rhythm component difficulty level. The rhythm selection is also weighted by its perceived difficulty. The more difficult the rhythm, the higher it is weighted in the selection. This ensures that difficult rhythms appear as the rhythm component level increases.

We further control the musicality of PiaNote's pieces by grouping certain rhythms. For example, eighth notes appear in groups of two. Similarly, sixteenth notes appear in groups of four. Dotted rhythms, such as a dotted quarter note or a dotted eighth note, are always followed by an eighth note or a sixteenth note, respectively. This prevents bizarre rhythm patterns, such as a dotted eighth note followed by a half note. Bizarre rhythms could drastically alter the difficulty of a piece. We also ensure that no rhythms span across measures. A half note or a dotted half note will never be placed on beat 4, for instance.

Lastly, PiaNote constructs the last measure such that the final rhythm is not less than a quarter note. We do this to give the piece a sense of closure.

#### 5.3.2.2 Constructing Pitches

After PiaNote selects rhythms, it assigns pitches to each rhythm. Pitch selection is difficult because we must keep interval difficulty level, hand positions, and musicality in mind.

PiaNote selects intervals according to the current interval difficulty level. For example, on an interval difficulty of 2, only the following intervals are accepted: 1sts, 2nds, 3rds, and 5ths. If the intervals component is a focus component, PiaNote prioritizes the most difficult interval. Using the previous example, if a user is focusing on intervals at a difficulty of 2, at least one 5th interval would be guaranteed to appear in the piece.

For simplicity, all PiaNote melodies do not stray from the five finger position. Once a hand is correctly placed at the start of a piece, it will not need to switch positions in order to play the melody. This constrains the amount of pitches that can be selected, simplifying both fingering notation and also interval selection. PiaNote chooses the hand position by selecting a starting pitch as well as a random fingering. With this information, the five possible pitches for a piece can be generated.

Hand position is also constrained by an octave. In the key of C, for example, hand positions must remain between C5 (middle C) and C6. In PiaNote, it is not possible to assign the right index finger to middle C when in the key of C, because this position strays beyond the octave C5-C6. This further simplifies music creation in order to prevent the left and right hand from colliding. Without octave constriction, the melodies may overlap. While this is possible in real world musical pieces, we avoid it in PiaNote to prevent further complexities in music generation.

PiaNote selects the first note of each measure based on the chord to make melodies more musical. The pitch is selected randomly from the set of three notes that make

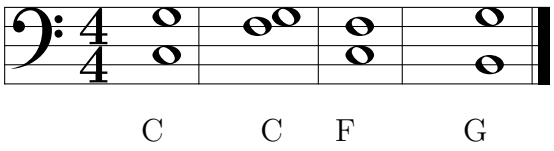



**Figure 5.10: PiaNote two-note harmony formations in the key of C. The letters below the staff refer to the note that the harmony is based upon.**

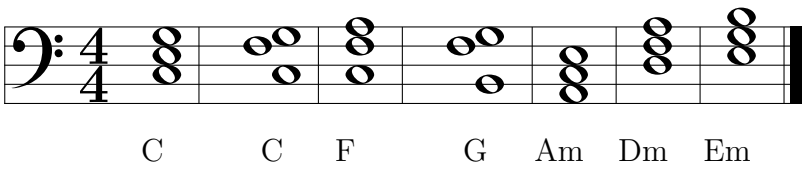



**Figure 5.11: PiaNote chord formations in the key of C. The letters below the staff refer to the note that the chord is based upon.**

up the chord's triad. For example, if the measure chord is C major, possible pitches for the first note are **C**, **G**, or **E**. PiaNote selects all other notes randomly based on the fingering position and the interval difficulty.

#### 5.3.2.3 Constructing Chords

Chord construction for the left hand is based on the song type. For a Simple Chord piece, PiaNote creates two note harmonies based on the measure chord. PiaNote creates three note chords for a Simple Full Chord Piece and Mixed Chord Piece. Figures 5.10 and 5.11 show the possible harmonies and chords for pieces in C major.

As seen in 5.11, PiaNote presents all chords such that users perform no hand jumps no larger than a third. This prevents added difficulty of left hand interval jumps. This approach also introduces complex chords such as six-four chords, dominant 7th chords, and suspended chords. PiaNote always presents the subdominant chord as a six-four chord, the dominant chord as a dominant 7th chord, and the tonic chord as a suspended chord if it occurs in measures three or four. In the key of C, these chords are is F, G, and C, respectively. This can also be seen visually in Figure 5.11.

### 5.3.3 The Use of Performance Statistics

PiaNote also uses performance statistics from the user model to aid in the exercise generation process. Performance statistics contain average accuracies of specific musical components, which can be used in various ways.

PiaNote's goal is to present pieces with components that are neither too easy nor too hard. We want to present specific components to a user that are performed with an accuracy near 75%. In an attempt to identify appropriate components, we pick the component whose average accuracy is closest to 75%.

For example, if the component difficulty for key signatures is 2, PiaNote chooses from the keys C, G and F. For a particular user, the average accuracies in these keys are 0.9, 0.72, and 0.65, respectively. PiaNote would select the key of G because its average accuracy is the closest to 75%. PiaNote uses the this method to select specific key signatures, time signatures, and song types, but uses performance statistics differently for rhythms.

PiaNote weights rhythm selection by difficulty and accuracy. The higher the accuracy of a rhythm, the more likely it will show up. PiaNote uses this method for rhythms because they are not a structural component of a piece of music, such as key signature, time signature, or song type. If a user is performing well on a difficult rhythm, we want to present the rhythm even more to make future pieces challenging.

# Chapter 6

# VALIDATION

PiaNote strives to present users with relevant sight-reading material by means of performance monitoring. On a broader level, PiaNote aims to provide an effective sight-reading experience for musicians.

To validate PiaNote, we ask a series of questions:

1. Does PiaNote adapt correctly based on a user's performance?

2. Is PiaNote effective in recognizing sight-reading difficulties?

3. Does PiaNote improve a user's sight-reading over time?

4. Does PiaNote improve a user's confidence in sight-reading?

5. Are PiaNote's lessons musical?

6. Is PiaNote useful for musicians overall?

In order to answer these questions, we conducted a user study to analyze PiaNote's potential. We hypothesized that PiaNote is an effective tool that improves a user's sight-reading confidence and abilities.

## 6.1 User Study

We evaluate PiaNote's success with a study conducted on students in Cal Poly's MU 163 course. We broke the study into three separate fifteen-minute sessions. In each session, we asked users to perform exercises that PiaNote presented. After each session, participants commented about their experience with PiaNote.

This experiment contained both an experimental group and a control group. The experimental group was assigned the full version of PiaNote, while the control group was assigned with a version of PiaNote that adjusts difficulty based on time, not human performance. We did this in an attempt to isolate PiaNote's difficulty adjustment feature and test its effectiveness in engaging users, improving sight-reading confidence, and improving sight-reading abilities. We refer to the control group application as the non-adaptive application.

After every three exercises, the non-adaptive application raises the difficulty. This program still monitors users and presents results, but performance statistics are disabled. As a result, further piece adjustments based on performance statistics are not executed, and users are provided with pieces according to difficulty alone. We designed the non-adaptive application to be similar to other sight-reading tools that have no knowledge of a user's abilities or performance.

Students were not aware about two different version of the application. The interfaces of both applications are identical.

### 6.1.1 User Background

MU 163 is Piano Skills III, the third piano course for music majors. Students in the course practice sight-reading with James Bastien's *Sight Reading, Level 2* [1]. Students use Cal Poly's on-campus piano lab for in-class activities. The piano lab is

a room filled with digital piano keyboards connected to touch-screen computers. All students in the course are familiar with other musical education programs such as MacGAMUT [3].

### 6.1.2 Study Limitations

Because the study is small and students spent different amounts of time with the tool, we have difficulty calculating truly meaningful results about user improvement. We still present this data in order to comment on PiaNote's potential.

Nine students participated in this study. Three students were assigned to the control group, so only six students were in the experimental group. Due to availability complications, not all students were able to attend three sessions. As a result, only four students participated in all three sessions, while six students participated in at least two sessions.

Unfortunately, not all control group students were able to make it to all sessions. Only one control student attended all three sessions. This makes it difficult to compare PiaNote to the non-adaptive application, as we hoped to compare the effectiveness of dynamic difficulty adjustment in improving sight-reading. While we cannot make any firm conclusions in our comparison of the tools, we still present data from the non-adaptive application in order to make preliminary inferences about PiaNote's effectiveness.

Because of the study complications, results focus more on PiaNote's ability to adjust difficulty, survey result data, and the qualitative analysis of PiaNote. Future research must be conducted to fully analyze PiaNote's effectiveness in improving sight-reading abilities.

## 6.2 User Study Results

Overall results suggest that PiaNote presents musical exercises that improve a user's confidence in sight-reading. We break the results into four sections. Section 6.2.1 explores both PiaNote's difficulty adjustments for individual user sessions and user feedback on adjustments. Section 6.2.2 analyzes whether sessions with PiaNote actually improve users' sight-reading. Sections 6.2.3 and 6.2.4 explore user feedback from the entire study.

### 6.2.1 Individual User Session

We present individual user sessions to exemplify that PiaNote adjusts difficulty while still actively challenging a user. Table 6.1 shows a snapshot of one user's session, which we call User A. Difficulty significantly decreases between songs 34 and 35 in reaction to the struggling performance at level $\langle$ 2, 2, 2, 2, 1$\rangle$. We see that after User A performs well in song 36, PiaNote increases difficulty back to the target level, $\langle$ 2, 2, 2, 2, 2$\rangle$.

We present a snapshot of another user's session, User B, in Table 6.2. This table shows PiaNote's ability to retreat to a previous difficulty if a user repeatedly struggles in the current level. If we look at songs 13 - 15, we notice that PiaNote focuses on key signature and rhythms. User B struggles significantly in rhythms three times, so PiaNote retreats to the previous level seen in song 12, with just a focus on rhythms. PiaNote eventually focuses on key signatures and rhythms again once User B performs well in song 17.

The last two columns of both tables display results from a survey that users took after each song. The survey was as follows:

1. How helpful was the piece to your learning? (0: Not Helpful, 10: Very Helpful)

**Table 6.1: Snapshot of User A's second session. PiaNote automatically adjusts levels based on component performance**

| Song | Level ⟨k, t, s, i, r⟩ | Focus | Accuracy ⟨k, t, s, i, r⟩ | Helpful (1-10) | Difficult (1-10) |
|---|---|---|---|---|---|
| 33 | ⟨2, 2, 2, 2, 2⟩ | {} | ⟨0.80, 0.73, 0.73, 0.91, 0.67⟩ | 5 | 5 |
| 34 | ⟨2, 2, 2, 2, 1⟩ | {} | ⟨0.67, 0.57, 0.57, 0.82, 0.47⟩ | 7 | 8 |
| 35 | ⟨1, 1, 1, 2, 1⟩ | {} | ⟨0.75, 0.71, 0.71, 0.75, 0.67⟩ | 5 | 5 |
| 36 | ⟨1, 1, 1, 2, 1⟩ | {} | ⟨1.00, 1.00, 1.00, 1.00, 1.00⟩ | 3 | 3 |
| 37 | ⟨2, 2, 2, 2, 2⟩ | {} | ⟨0.77, 0.73, 0.73, 0.78, 0.68⟩ | 8 | 8 |
| 38 | ⟨2, 2, 2, 2, 1⟩ | {} | ⟨0.87, 0.80, 0.80, 1.00, 0.73⟩ | 5 | 5 |
| 39 | ⟨2, 2, 2, 2, 1⟩ | {} | ⟨0.93, 0.93, 0.93, 1.00, 0.93⟩ | 5 | 5 |
| 40 | ⟨2, 2, 2, 2, 2⟩ | {} | ⟨0.95, 0.81, 0.81, 0.94, 0.67⟩ | 5 | 5 |
| 41 | ⟨2, 2, 2, 2, 1⟩ | {} | ⟨1.00, 1.00, 1.00, 1.00, 1.00⟩ | 3 | 3 |
| 42 | ⟨2, 2, 2, 2, 2⟩ | {} | ⟨0.94, 0.94, 0.94, 1.00, 0.94⟩ | 5 | 5 |
| 43 | ⟨3, 2, 2, 2, 2⟩ | {k} | ⟨1.00, 0.96, 0.96, 1.00, 0.93⟩ | 5 | 5 |
| 44 | ⟨2, 3, 2, 2, 2⟩ | {t} | ⟨0.79, 0.79, 0.79, 0.90, 0.79⟩ | 7 | 8 |
| 45 | ⟨2, 3, 2, 2, 2⟩ | {t} | ⟨0.94, 0.74, 0.74, 0.92, 0.53⟩ | 8 | 8 |
| 46 | ⟨2, 3, 2, 2, 1⟩ | {t} | ⟨1.00, 0.95, 0.95, 1.00, 0.91⟩ | 5 | 5 |
| 47 | ⟨2, 3, 2, 2, 2⟩ | {t} | ⟨0.92, 0.88, 0.88, 1.00, 0.85⟩ | 5 | 5 |
| 48 | ⟨3, 3, 2, 2, 2⟩ | {k t} | ⟨1.00, 0.78, 0.78, 1.00, 0.56⟩ | 5 | 5 |
| 49 | ⟨3, 3, 2, 2, 1⟩ | {k t} | ⟨0.73, 0.73, 0.73, 0.86, 0.73⟩ | 2 | 2 |

2. How difficult was this piece? (0: Too Easy, 5: Perfect Difficulty, 10: Too Hard)

The results of this survey are displayed in the *Helpful* column and the *Difficult* column, respectively. Our goal is to provide users with pieces that are perceived as a

**Table 6.2: Snapshot of User B's second session. PiaNote automatically adjusts levels based on component performance**

| Song | Level ⟨k, t, s, i, r⟩ | Focus | Accuracy ⟨k, t, s, i, r⟩ | Helpful (1-10) | Difficult (1-10) |
|---|---|---|---|---|---|
| 10 | ⟨1, 2, 2, 2, 1⟩ | {t s i} | ⟨0.91, 0.91, 0.91, 0.86, 0.91⟩ | 5 | 5 |
| 11 | ⟨2, 2, 2, 2, 1⟩ | {k t s i} | ⟨0.91, 0.91, 0.91, 1.00, 0.91⟩ | 5 | 5 |
| 12 | ⟨1, 1, 1, 1, 2⟩ | {r} | ⟨1.00, 0.95, 0.95, 1.00, 0.91⟩ | 7 | 5 |
| 13 | ⟨2, 1, 1, 1, 2⟩ | {k r} | ⟨0.64, 0.57, 0.57, 0.64, 0.50⟩ | 8 | 6 |
| 14 | ⟨2, 1, 1, 1, 2⟩ | {k r} | ⟨0.64, 0.59, 0.59, 0.64, 0.55⟩ | 7 | 6 |
| 15 | ⟨2, 1, 1, 1, 2⟩ | {k r} | ⟨0.83, 0.71, 0.71, 0.83, 0.58⟩ | 7 | 6 |
| 16 | ⟨1, 1, 1, 1, 2⟩ | {r} | ⟨0.92, 0.81, 0.81, 0.92, 0.69⟩ | 7 | 6 |
| 17 | ⟨1, 1, 1, 1, 2⟩ | {r} | ⟨0.94, 0.94, 0.94, 0.94, 0.94⟩ | 7 | 6 |
| 18 | ⟨2, 1, 1, 1, 2⟩ | {k r} | ⟨1.00, 1.00, 1.00, 1.00, 1.00⟩ | 8 | 6 |
| 19 | ⟨1, 2, 1, 1, 2⟩ | {t r} | ⟨0.92, 0.83, 0.83, 0.92, 0.75⟩ | 7 | 7 |
| 20 | ⟨1, 2, 1, 1, 2⟩ | {t r} | ⟨1.00, 1.00, 1.00, 1.00, 1.00⟩ | 7 | 5 |

5-7 difficulty. This way, PiaNote presents pieces that are challenging but still within a user's abilities.

We see that User A rates song 36 as a 3 in difficulty, song 37 as an 8, and song 39 as a 5. Because PiaNote is adjusting difficulty based on performance, we see that it aims to provide User A with a song in the 5-7 difficulty range. The average perceived difficulty of the pieces presented in User A and B's snapshot is 5.29 and 5.73, respectively.

It is important to address the helpfulness ratings in both snapshots. The target helpfulness rating is 7-10, as we aim to always provide helpful lessons. We notice

that User A's helpfulness rating is the highest when the pieces are perceived as very difficult. In contrast, User B finds a milder difficulty still helpful towards learning.

It is difficult to measure the target difficulty that maximizes user satisfaction, because our sample size is small. Nevertheless, feedback from User A and B may suggest that PiaNote should use both performance accuracies and user opinion in order to adjust difficulty effectively. This way, users are still able to work at their own pace while also receiving satisfying sight-reading exercises.

### 6.2.2 Pre-test and Post-test

In an attempt to further analyze PiaNote's capabilities, we provided participants with a pre-test and a post-test at the beginning and the end of the study. We gave the pre-test to users on their first session with PiaNote. We gave the post-test to users who attended the third session of the study.

The pre-test and post-test contain three pieces at various difficulties. If we view a level as the vector of components, $\langle$k, t, s, i, r$\rangle$, the difficulty of the three pieces are $\langle$1,1,1,1,1$\rangle$, $\langle$2,2,2,2,2$\rangle$, and $\langle$3,3,3,3,3$\rangle$, respectively.

The exercises in the pre-test and post-test are pre-generated. All users received the exact same test. We did this to control the experiment, as randomly generated pieces could alter difficulty slightly and skew results. Contents of the pre-test and the post-test can be found in Appendix A and Appendix B.

We originally hypothesized that performances on the post-test would be higher in the experimental group than those of the control group. Because our sample size for the control group is only one, due to study complications, we could not effectively test this hypothesis. Instead, we analyze whether repeated sessions with PiaNote provide any improvement at all.

**Table 6.3: Pre-test and post-test average accuracies. Data is collected from five participants.**

| Test | Piece 1 | Piece 2 | Piece 3 | Average |
|---|---|---|---|---|
| Pre-test | 56.36% | 63.25% | 57.83% | 59.15% |
| Post-test | 96.92% | 62.03% | 56.09% | 71.68% |

We present average pre-test and post-test scores of experimental group participants in Table 6.3 and Figure 6.1. We see that while performance on the first post-test piece is much higher than that of the pre-test, the average performance on second and third post-test piece has not significantly changed. This makes sense because difficulty in PiaNote is adaptive, so many users were never presented with pieces as difficult as post-test pieces 2 and 3. We recognize that in future studies, extra intermediate difficulties and exercises should be generated for the tests in order to present more accurate results.

It is difficult to draw any conclusions about PiaNote's ability to improve sight-reading abilities because this data is averaged from only five participants. However, we see an average performance increase of 12.53% between the pre-test and post-test. A larger study should be conducted in order to completely assess PiaNote's ability to improve a musician's sight-reading.

### 6.2.3 User Survey Results

Participants answered six questions after each session with PiaNote. The six questions are as follows:

1. How effective was the tool in assessing your sight-reading difficulties? (1-10, 1=Not Effective at All, 10=Very Effective)

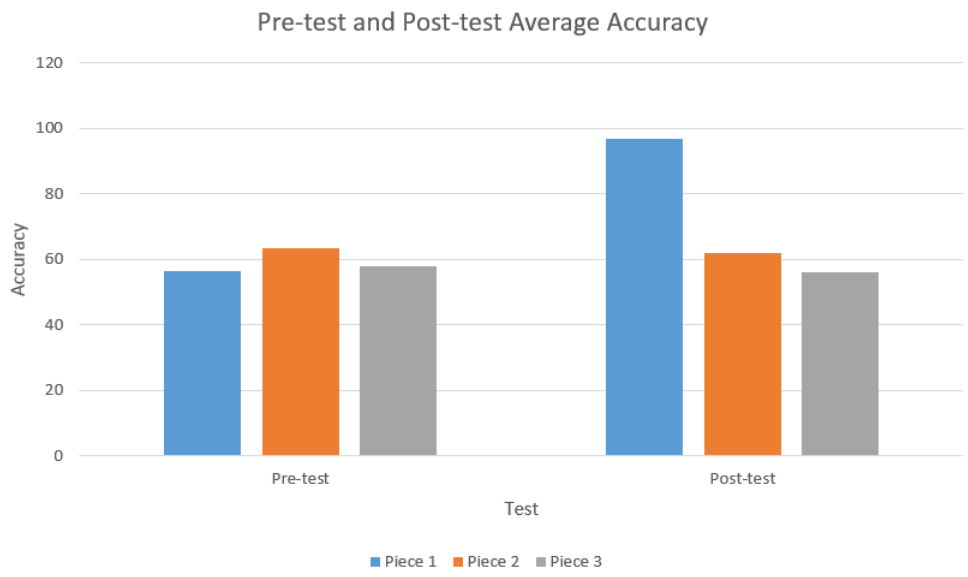



**Figure 6.1: Pre-test and post-test average performance.**

2. How musical were the pieces presented to you? (1 - 10, 1=Not Musical at all, 10=Very Musical)

3. To what extent has your confidence in sight-reading improved after using the tool today? (1-10, 1=No improvement, 10=A lot of improvement)

4. Were there any situations where the tool presented pieces that were not helpful to your learning? If so, please explain.

5. What was your overall experience with the tool today? (1-10, 1=Terrible, 10=Fantastic)

6. Please comment on your experience with the tool.

**Table 6.4: Average survey Ratings grouped by session.**

| Survey Category | Session 1 | Session 2 | Session 3 |
|---|---|---|---|
| Effectiveness | 7.33 | 8.67 | 8.33 |
| Musicality | 8.33 | 9 | 8.67 |
| Sight-Reading Confidence | 4.67 | 6 | 7 |
| Overall Experience | 8.33 | 8 | 8 |

We chose to include qualitative questions in order to assess what users like and dislike about the tool. We grouped surveys based on a user's session number. After grouping surveys, we averaged all ratings which are presented in Table 6.4 and Figure 6.2. With the exception of overall experience, all average scores generally rise over time, suggesting that extended use of the tool improves recognition of user difficulties, musicality of the lessons, and user confidence in sight-reading.

In comparison to other survey rankings, participants' overall confidence dramatically increased from 4.7 during the initial session, to 7 in the third and final session. This could indicate that over time, users of PiaNote become more confident in sight-reading.

We also present the survey results from the user in the control group in Figure 6.3. We notice that confidence level ratings are significantly lower than those in the adaptive tool. This suggests that users feel more confident in sight-reading when using the adaptive tool.

In Figure 6.2, it is interesting that musicality remains fairly stable. Musicality scores from session one may be lower because easier lessons do not contain chords in the left hand. Instead, both hands play separate melodies. A lack of harmony may make the melodies less apparent, possibly affecting musicality. There also may simply not be enough data to effectively measure the musicality of the exercises. Data in

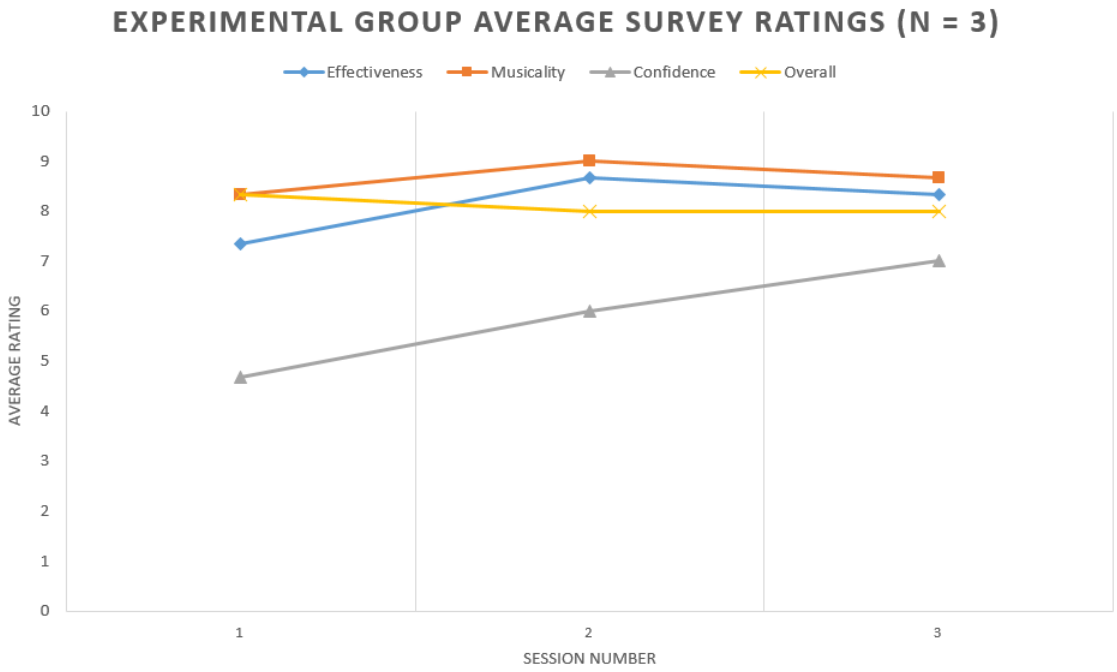



**Figure 6.2: Average survey ratings grouped by session.**

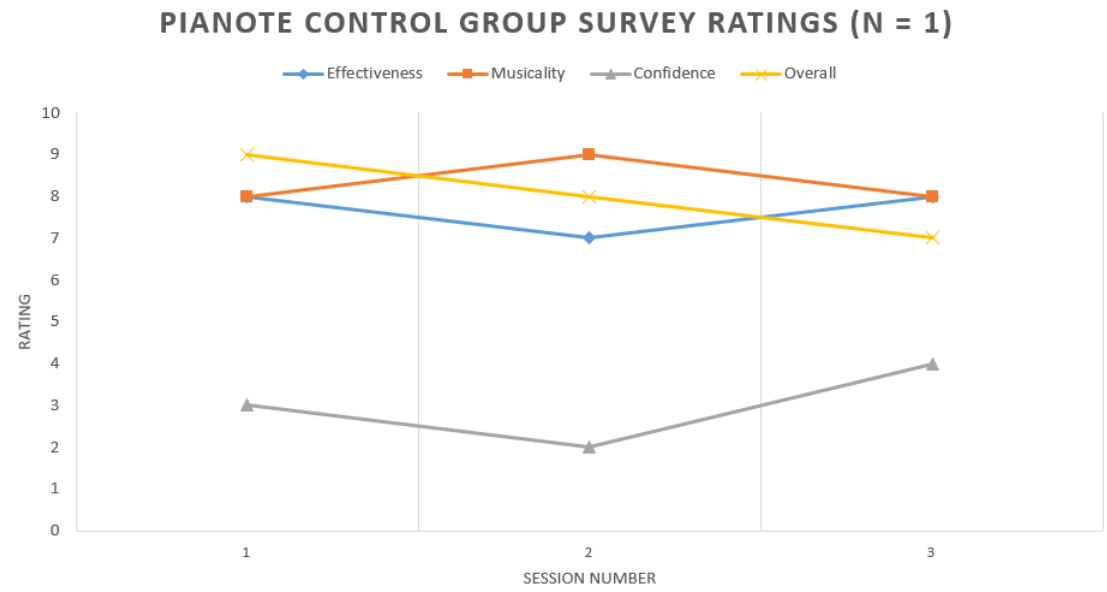



**Figure 6.3: Average control group survey ratings grouped by session.**

Table 6.4 show that musicality remains above 8, suggesting that PiaNote's exercises are musical regardless of difficulty level.

Although the average ratings increase or decrease over the course of the sessions, it is worth noting that all average ratings are relatively high. The high ratings suggest that PiaNote has the potential to effectively recognize student difficulties, provide musical exercises, and improve sight-reading confidence.

### 6.2.4 Qualitative Feedback

A lot of the qualitative survey feedback was helpful in identifying PiaNote's strengths and weaknesses. All survey responses can be found in Appendix C.

PiaNote receives mostly positive feedback. Users mention that it is a helpful tool, with some comments noting that it provides a fun environment to learn sight-reading, actively challenges users, allows users to find areas where they need improvement, and provides a more engaging experience than a sight-reading book.

Users also describe some of PiaNote's weak points. One user comments, "a lot of the time it will mark a rhythm as wrong because it only slightly gets cut off". This user notices that PiaNote's monitoring system may be too strict, an issue that we address in Sections 5.1.5 and 5.1.6. It appears that while PiaNote attempts to be lenient on rhythm monitoring, it is still too strict when capturing smaller rhythms, such as eighth notes. Further work must be done to improve PiaNote's monitoring system, as this portion of the program is essential in correctly adjusting difficulty.

Some users also mention that PiaNote jumps between pieces that are too easy, and pieces that are too difficult. This could be related to a problem called the "Rubber Band" effect, a concept often addressed in regards to dynamic difficulty adjustment [13]. The rubber band effect occurs when the tool fails to provide an appropriate difficulty for the user and makes the difficulty too hard when a user performs well,

and too easy when a user is struggling. This causes the difficulty to continually oscillate, and may distract and upset users. The fact that some users mention the existence of a rubber band effect in PiaNote suggests that either difficulty adjustment may be too sensitive, or that PiaNote may not provide enough intermediate levels.

When we suggest that difficulty adjustment may be too sensitive, we mean that PiaNote may tweak difficulty so frequently that it is not accurately monitoring a user's abilities, but merely reacting to independent performances. Although PiaNote follows a roughly linear difficulty progression, it may react too frequently without allowing users to adjust to the current difficulty. Other users comment that occasionally PiaNote adjusts difficulty too much when they happen to fumble on just one performance. This was problematic for some users that simply made a mistake on one piece, while performing well on others. This suggests that PiaNote should conduct more analysis on a user's progress when adjusting difficulty, instead of making adjustments solely based on the current performance. At the very least, PiaNote should be modified to take user feedback into account when adjusting difficulty.

Another issue that may introduce the rubber band effect is the lack of sufficient difficulty levels. PiaNote contains some large gaps between difficulties, most notably in song type. Song type difficulty progresses from separate hand melodies to pieces with two note harmonies in the left hand. This difficulty gap may simply be so large that some users experience rapid difficulty adjustments. Extra intermediate difficulties were not included in PiaNote due to time constraints, but this addition could be one simple way to fix drastic difficulty adjustment.

### 6.2.5 Results Summary

While complications arose in the study, the data gathered suggests that PiaNote has the potential to become an application that can effectively aid pianists in sight-

reading. Further research should be conducted to make firm conclusions about PiaNote, especially its ability to improve a musician's sight-reading abilities. It is unfortunate that none of the questions posed about PiaNote could be answered with clarity. However, current data suggests that all answers to our questions may be "yes".

# Chapter 7

# FUTURE WORK

While PiaNote is a functional sight-reading application, further research can be done to improve PiaNote and validate its effectiveness. This chapter outlines various areas of research that could be explored.

## 7.1 Performance Monitoring

PiaNote's monitoring system is not perfect and still has the potential to interpret intended rhythms incorrectly. User feedback suggests that PiaNote struggles most with smaller rhythms such as eighth notes and sixteenth notes. Currently, PiaNote employs an epsilon value when matching all rhythms. For example, if a performance duration is within a 40% error, it is matched with the current rhythm.

One way to improve monitoring is to study average key release times relative to hit times of successive notes. If this is analyzed for each rhythm, it may be possible to accurately match performance durations to smaller rhythms. If the performed release time is relatively close to the expected release time, we could match performance durations more accurately.

Another suggestion is to modify the leniency of the rhythm monitoring based on difficulty. For easier difficulties, monitoring would be very lenient, while at harder difficulties monitoring would be stricter. This could introduce more challenge and also help users develop more precise rhythm skills, without causing frustration early on.

## 7.2 Difficulty Adjustment

PiaNote's dynamic difficulty adjustment was created with sight-reading in mind. It would be interesting to develop different methods of difficulty adjustment in order to compare their relative effectiveness. One option could be to remove all linear difficulty progression and rely on performance statistics alone to generate pieces of music. This could result in an application that is more user tailored, as some of our difficulty rankings are subjective. Currently, we rank $\frac{2}{4}$ time at a higher difficulty than $\frac{3}{4}$ time. It may be the case that some individuals find $\frac{3}{4}$ time harder than $\frac{2}{4}$. If PiaNote used performance statistics alone, it may be able to always present users with pieces that challenge them in their own ways.

Another option is to add the ability to adjust difficulty based on user feedback ratings. As we notice in Section 6.2.1, some users like to be challenged more than others. As sight-reading can be a stressful activity on its own, applications and games must be careful to not easily frustrate users any further. Common recommendation system approaches could be used to adjust difficulty based on user ratings.

Lastly, the easiest option to improve PiaNote's difficulty adjustment algorithm is to simply add more intermediate levels. Currently we receive some user feedback that suggests that the application is unable to find a difficulty that meets their needs. To solve this, extra song types could be added to the application in order to provide a more gradual difficulty progression. At the same time, it is also important to keep advanced musicians in mind that would still like to progress with rapid difficulty progression.

## 7.3 PiaNote Validation

Unfortunately, many complications arose in the initial study of PiaNote. Due to these complications, it is difficult to draw any clear conclusions about PiaNote's effectiveness, especially its dynamic difficulty adjustment algorithm. Another study should be conducted that contains far more participants and more comprehensive pre-tests and post-tests in order to test whether PiaNote improves a musician's ability to sight-read. Further study could aid in the development of more effective sight-reading applications than those that exist today.

# Chapter 8

# CONCLUSION

In this thesis we presented PiaNote, a sight-reading application that generates pieces based on human performance. We were able to successfully create an application that could algorithmically generate sheet music, monitor user performance in various musical components, and dynamically adjust exercise difficulty. To validate this application, we conducted a user study where users performed lessons that PiaNote presented and also provided feedback. Complications with the study hindered results, especially in measuring PiaNote's ability to improve a musician's sight-reading. However, the data gathered does provide interesting preliminary findings about PiaNote as a whole.

PiaNote has the potential to provide users with a sight-reading environment that can constantly provide appropriate musical exercises, improve a musician's sight-reading abilities, and improve a musician's confidence in sight-reading. We believe that future studies will provide more accurate results and facilitate further development of effective sight-reading applications.

# BIBLIOGRAPHY


[1] J. Bastien. *Sight Reading, Level 2.* Neil A. Kjos Music Co., San Diego, California, 1976.

[2] J. Biles. Genjam: A genetic algorithm for generating jazz solos. In *Proceedings of the International Computer Music Conference*, pages 131–131. INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 1994.

[3] A. K. Blombach. *The Complete User Guide to MacGAMUT 6.* MacGAMUT Music Software International, Columbus, OH, 2008.

[4] Connect For Education Inc. On music dictionary, 2015. [Online; accessed November 29, 2015. http://dictionary.onmusic.org/terms/2278-music].

[5] D. Cope. The well-programmed clavier: Style in computer music composition. *XRDS*, 19(4):16–20, June 2013.

[6] R. B. Dannenberg, M. Sanchez, A. Joseph, R. Joseph, R. Saul, and P. Capell. Results from the piano tutor project. In *Proceedings of the Fourth Biennial Arts and Technology Symposium*, pages 143–150, 1993.

[7] M. Deal. Midi.js, 2013. [Online; accessed November 29, 2015. https://github.com/mudcube/MIDI.js/].

[8] M. Edwards. Algorithmic composition: Computational thinking in music. *Commun. ACM*, 54(7):58–67, July 2011.

[9] A. Ghassaei. What is midi, 2013. [Online; accessed November 29, 2015. http://www.instructables.com/id/What-is-MIDI/].

[10] GraceNotes, LLC. Sight Reading Factory, 2015. [Online; accessed November 29, 2015. https://www.sightreadingfactory.com/].

[11] M. Grinberg. *Flask Web Development: Developing Web Applications with Python*. " O'Reilly Media, Inc.", 2014.

[12] A. Hronco. Making music in the browser web midi api. `https://www.keithmcmillen.com/blog/making-music-in-the-browser-web-midi-api/`, 2015.

[13] R. Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM, 2005.

[14] H. Järveläinen. Algorithmic musical composition. In *Seminar on content creation Art@ Science*, 2000.

[15] M. Kennedy and J. Bourne. *The Concise Oxford Dictionary of Music*. Oxford University Press, 2004.

[16] J. R. Luce. Sight-reading and ear-playing abilities as related to instrumental music students. *Journal of Research in Music Education*, pages 101–109, 1965.

[17] M. Papamanolis. Presto Keys, 2015. [Online; accessed November 29, 2015. http://www.prestokeys.com/].

[18] B. Purse. *The PrintMusic! primer: mastering the art of music notation with Finale PrintMusic!* Hal Leonard Corporation, 2003.

[19] E. S. Ristad and P. N. Yianilos. Learning string-edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5):522–532, 1998.

[20] P. Rosen. abcjs, 2014. [Online; accessed January 19, 2016. https://github.com/paulrosen/abcjs].

[21] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[22] S. W. Smoliar, J. A. Waterworth, and P. R. Kellock. pianoforte: A system for piano education beyond notation literacy. In *Proceedings of the Third ACM International Conference on Multimedia*, MULTIMEDIA '95, pages 457–465, New York, NY, USA, 1995. ACM.

[23] Y. Tayama, R. Kato, and K.-i. Okada. Triage training system: Adjusting the difficulty level according to user proficiency. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, MUM '15, pages 139–147, New York, NY, USA, 2015. ACM.

[24] C. Walshaw. Abc2mtex: An easy way of transcribing folk and traditional music, version 1.0. *University of Greenwich, London*, 1993.

[25] C. Walshaw et al. A statistical analysis of the abc music notation corpus: exploring duplication. 2014.

[26] Wikipedia, the Free Encyclopedia. A simple two-state markov chain, 2013. [Image online; accessed November 29, 2015. https://en.wikipedia.org/wiki/Markov_chain/media/File:Markovkate_01.svg].

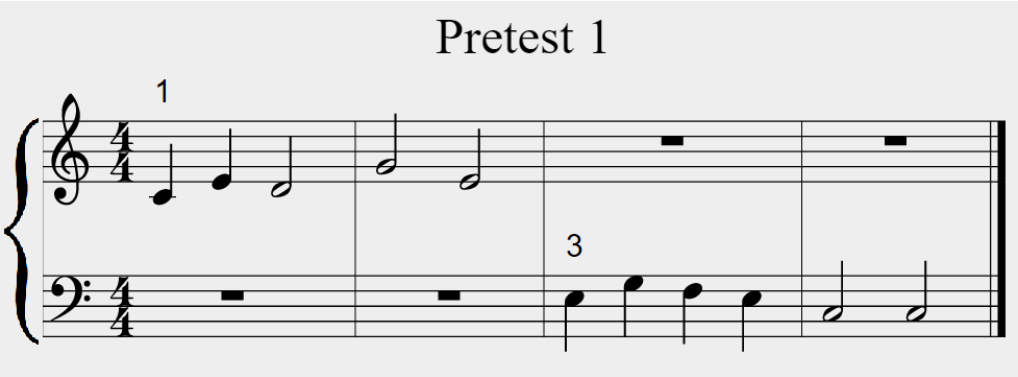
[27] C. Wilson and J. Kalliokoski. Web midi api. Technical report, W3C, 2015. https://www.w3.org/TR/webmidi/.

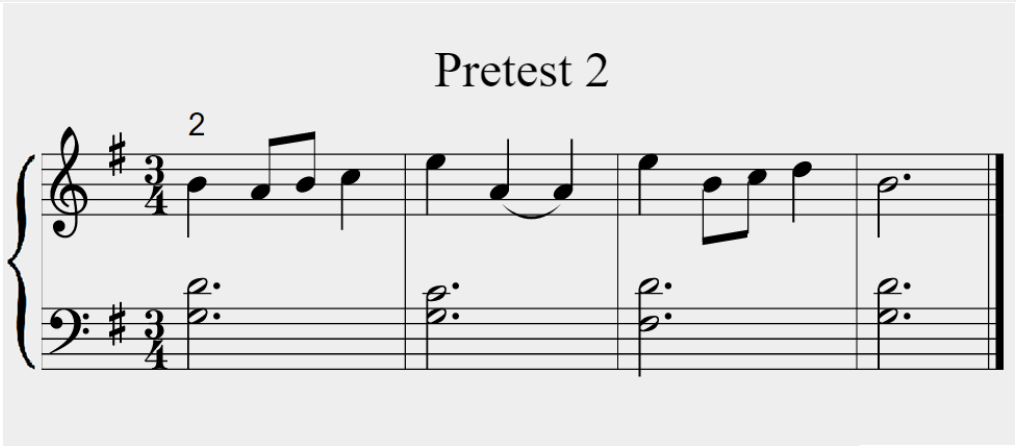
[28] A. Zook and M. O. Riedl. A temporal data-driven player model for dynamic difficulty adjustment. In *AIIDE*. Citeseer, 2012.

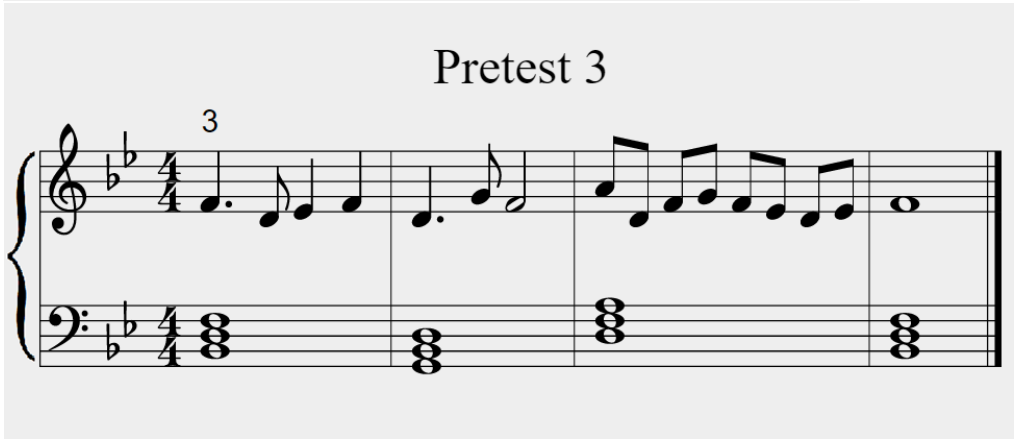
APPENDICES

Appendix A

PRE-TEST

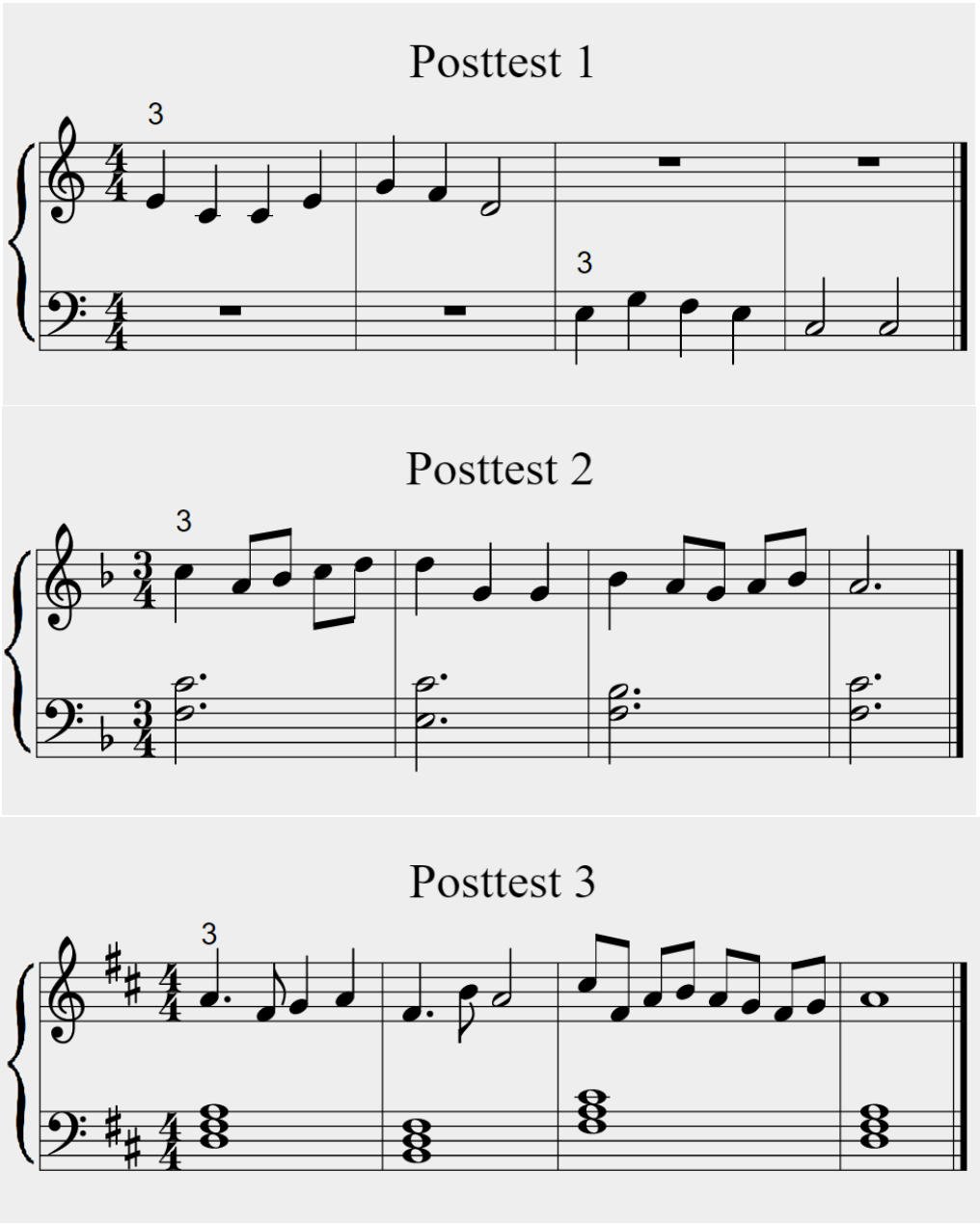


Appendix B

POST-TEST



Appendix C

USER SURVEY

1. How effective was the tool in assessing your sight reading difficulties? (1-10, 1=Not Effective at All, 10=Very Effective)
2. How musical were the pieces presented to you? (1 - 10, 1=Not Musical at all, 10=Very Musical)
3. To what extent has your confidence in sight reading improved after using the tool today? (1-10, 1=No improvement, 10=A lot of improvement)
4. Were there any situations where the tool presented pieces that were not helpful to your learning? If so, please explain.
5. What was your overall experience with the tool today? (1-10, 1=Terrible, 10=Fantastic)
6. Please comment on your experience with the tool

| Session | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|---|
| 1 | 7 | 10 | 7 | No | 10 | I thought it was very helpful and I would love to use it more in the future. |
| 1 | 5 | 5 | 2 | Tool kept bouncing back and forth from really simple, to slightly too difficult. | 5 | I enjoyed it. It's a fun way to improve! |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 8 | 7 | 7 | Some pieces were much too easy, but could be helpful to people just learning keys on a piano as they only had 2 keys. | 9 | It was very easily understood. |
| 1 | 10 | 10 | 5 | N/A | 10 | I already have a bit of piano experience, but it is an awesome tool! |
| 1 | 7 | 6 | 6 | No | 10 | The metronome couldn't go any slower than 80 bpm which wasn't terribly fast for me however there were times where I wish I could go a little slower. |
| 1 | 8 | 5 | 7 | There were no pieces that were unhelpful to my learning. | 9 | It was very helpful in sight-reading, both challenged me and reassured me that I'm not a bad musician/piano player. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 10 | 10 | 7 | No | 10 | My experience with the tool was good, but a lot of the time it will mark a rhythm as wrong because it only slightly gets cut off. For example if you don't hold an eighth note for its full value it will be wrong but you can't hold it without missing the next note. |
| 2 | 8 | 7 | 5 | Nope. | 7 | It is a very fun way to learn/practice sight-reading. Only glitch I noticed is when you have the C + B in the bass clef, the line doesn't get put in the right place (only when it's a whole note). |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 4 | 8 | 4 | At first, it continuously gave pieces that were much too hard and I just fumbled through them. Then it gave much too easy pieces with only 2 or 3 notes. | 7 | There were technical difficulties at the beginning with the piano and the computer not being connected so a few of the songs went completely unplayed. |
| 2 | 8 | 10 | 6 | No. | 7 | I like the tool a lot I think it was very helpful and it let me understand where I need improvement. |
| 2 | 10 | 8 | 8 | Nope. | 10 | The tool was very helpful and great practice. I feel that for me it was more engaging than going through a sight-reading book. |
| 3 | 7 | 6 | 5 | Only after making a bunch of mistakes in one piece, it would get a little too easy. | 6 | It is a really fun way to improve my sight-reading skills. |
| 3 | 10 | 10 | 10 | No. | 10 | Super program, it works very well, congrats. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 8 | 10 | 6 | Yes. It seemed like it kept giving me the same type of pieces each time. | 8 | I like this tool a lot. I think it will be very helpful to music students. |

Table C.1: User survey responses for users of the adaptive sight-reading tool.