Piano Training Simulator
Final Report

Prepared by
Abel Abraham, Beyza Soylu, Edward Plesa, Ryan Jasiak

CS 440
University of Illinois Chicago

April 2022

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

As one could probably guess, it is hard to find a piano just anywhere, and learning piano without an instructor is not easy. Nowadays, there are plenty of people, including professionals, who struggle playing certain elements of a song and need a way to hone their skills. This piano training simulator helps in providing meaningful analysis of users' techniques. Moreover, this application helps train users of all skill levels (even those who have prior experience) how to play the piano and increase their playing ability.

This project has two main parts. The first part is a physical keyboard that plugs into a computer. It will simulate the look and feel of actual piano keys. The input from the keyboard will be interpreted by the second part of the project: a software application installed on a user's computer. This software application will virtualize sound that mimics an actual piano and outputs it to the headphones or speakers. It will also analyze the user's input compared to what song they are supposed to be playing in real time. It will give live feedback of mistakes and give overall results to help the user improve and practice [1] Section 1: Project Overview.

2 Project Domain

The domain of the project falls under Artificial Intelligence and is meant for any person(s) who want to learn to play piano. The project relies mainly on the analysis of a player's performance during song playing to recommend where a user should practice to make improvements in their future performances. Naturally, the overall product is intended for anyone with an inclination towards learning piano, regardless of age or skill level.

3 Relationship to Other Documents

To maintain the original vision of the project, this document will heavily reference the work done by Group 19's "Piano Training Simulator" development document from Fall of 2018 which includes the Project Description Section, Project Requirements Section, and the Project Design Section.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

Further discussion of the Piano Training Simulator will require some additional concrete definitions for terminology:

- MIDI: The Industry Standard for the communication of musical note messages being sent too and from midi instruments. A MIDI message generally includes information regarding event tick, note, strike velocity and oftentimes more information.

- Event Tick: Determines *when* a MIDI event should be executed in the context of a MIDI file

  Other necessary key terms will be referenced from [1] Section 7a: Definition of Key Terms.

4b UML and Other Notation Used in This Document

Necessary UML and other notation will be referenced from [1] Section 7b: UML and Other Notation Used in This Document.

4c Data Dictionary for Any Included Models

The Data Dictionary for any included models will be referenced from [1] Section 7c: Data Dictionary for Any Included Models.

- .midi Files: A file containing MIDI data for a song. The application plays songs by using .midi files. The user can add .midi files for any song they'd like to practice, and can set the bpm to whatever they'd like!
- User Data: Usernames passwords, and user training logs are stored on a server.

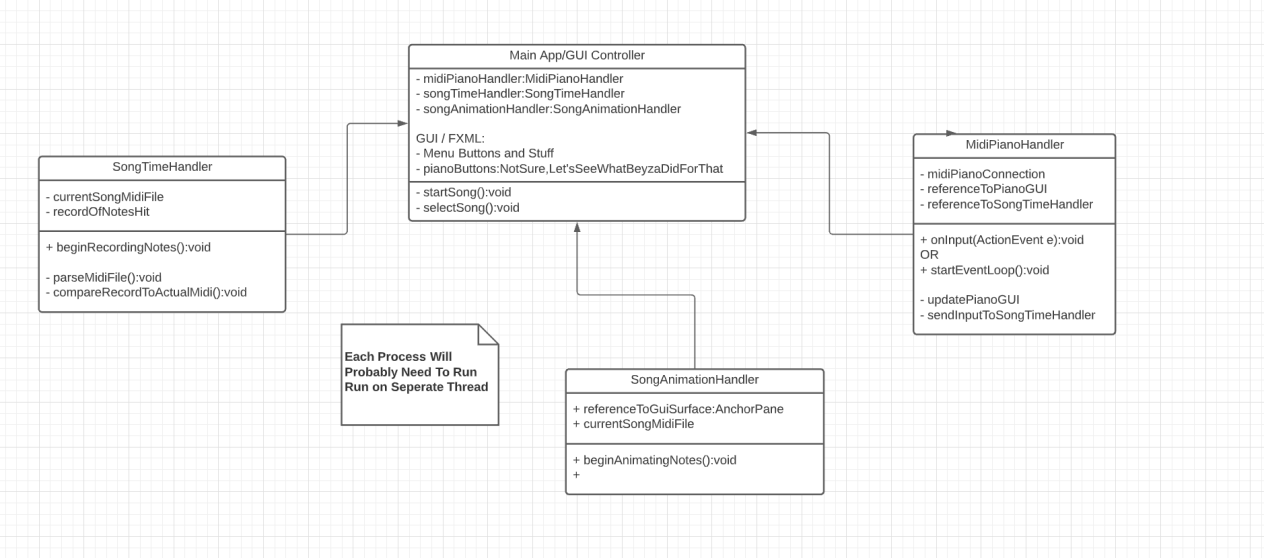
II Project Deliverables

5 First Release

The project's first demo was released February 25, 2022. This demo consisted of a working program that ran from the command line to open a GUI. The overall interface was very crude in terms of style which contained only a single scene. The scene contained a full sized piano with 7 octaves along the bottom of the screen. Additionally, there were 2 buttons, one for playing the selected song and the other for ending the playing song. The songs were only able to be changed by hard coding it in the software. The MIDI keyboard was capable of being recognized by the software but did not output any sound when pressing the keys. The alignment of the falling notes were also not fully implemented at this point, making it difficult to know which piano key should be used. The scoring system for playing correct/ incorrect notes was also not finished.

6 Second Release

Figure 1 - Main Application Subsystem Class Diagram



The project's second demo was released April 1, 2022. The demo remained the same in that the main GUI was launched from the command line. This demo's GUI consisted of 3 main scenes: one for user login, one for loading songs from project directory dynamically, and the last scene was the same main piano program from the first release, albeit, with more functionality and features. The login screen consisted of two text boxes allowing a user to enter a user name and password to press the login button.There were no security checks to see if a user was a registered member of the program, meaning you could login without any necessary credentials. The next scene was the song loading scene. Here, a user could press a button which will load all the songs found in the main software directory. Every song will be its own button which passes the necessary MIDI information to the main application scene. Finally, the main piano application scene remained nearly the same visually. There were 3 more buttons added: one for switching users, one for picking a new song, and the other for loading a MIDI keyboard to the application. The falling notes alignment was greatly improved from the previous demo, to where a user could tell which piano note should be played. The demo also implemented the processing of the piano notes leading to the correct audio to be played depending on the notes pressed on the keyboard or on screen.

7 Comparison with Original Project Design Document

The full project, as described by the previous group, wanted an all in one application to replace the need of tutors, guides, etc. The vision they had for the GUI necessitated having sheet music with a slider to show at which point the song is currently playing. They also had additional elements regarding sections for instructions about exercises, overall performance, and level of completion [1] Section 25f: User Interface. The document also mentioned having subsystems for managing user data, music sheet data, and training data (performance analysis).

Our final prototype implemented many of the key features outlined in the previous document. To begin, we have created the core functionality and framework for the user login system. A user is able to enter credentials into our prototype, however, it will not be checked against a database of legitimate users meaning there is no personal profile information or user data. Next, we implemented a way for a user to dynamically load music for use in our software. In the main application directory there is an area where a user can easily build a local database of songs. However, at this point, there is no remote database where a user can pick a song from. Our main application UI contains the virtual representation of a full sized keyboard and displays which key is pressed when the user pressed a key on the MIDI keyboard. There are options (buttons) for playing the current song, ending the current song, going back to the song list, connecting a MIDI keyboard, and switching users. This is slightly different as the mock GUI from the development guidelines have these features hidden in a menu bar. Our biggest change is that we replaced the sheet music that is displayed when a song is playing, to a version in which the piano notes are sliding down tracks to their respective keys. The keys also produce their own note when pressed on the connected MIDI keyboard. Finally, there is no implementation regarding level of song completion, overall performance, and on screen exercise instructions present in this prototype.

Figure 2 - Rough Low-Fidelity UI Mockup [1] Section 25f: User Interface

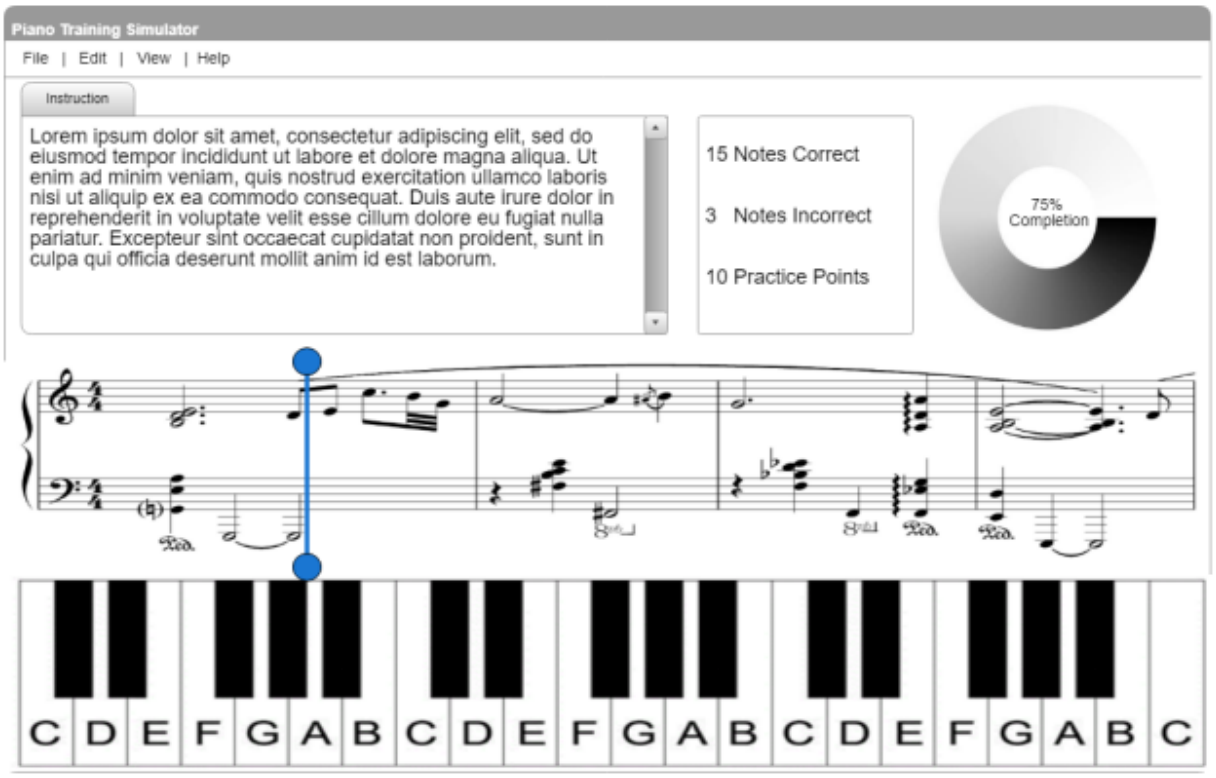
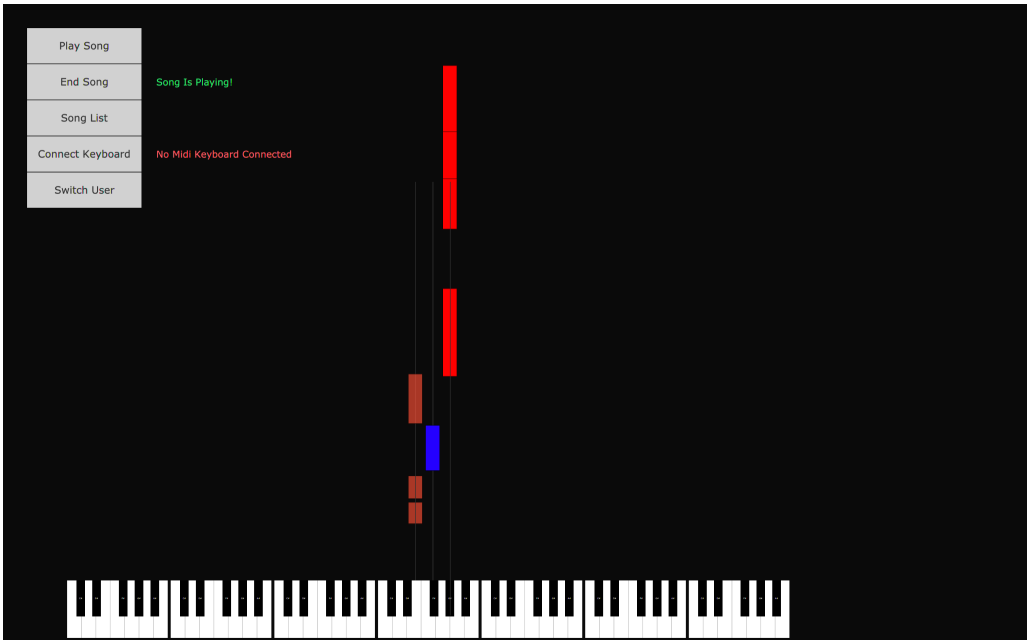


Figure 3 - Current State of Prototype UI



III Testing

8 Items to be Tested

- #470: MIDI Controller Connection
- #471: Note Sliding Algorithm
- #472: .midi File Parsing Algorithm
- #473: Song Scoring Algorithm

9 Test Specifications

470# - MIDI Controller Connection

Description: This test covers the reliability and usability of the connection to a MIDI Controller.

Items covered by this test: Connect to MIDI Controller

Requirements addressed by this test: Ability for the user to use an external keyboard.

Environmental needs:

Hardware Requirements: MIDI Controller

Software Requirements: Application to test connectivity

Intercase Dependencies:

Tests depending on this:

- Song scoring Test

Test Procedures: Test will be performed by writing code to print a value whenever a key is pressed on the MIDI Controller.

Input Specification: User Hitting Key

Output Specifications: Message will print in the console.

Pass/Fail Criteria: If the correct value is printed for each key press, test passes.

471# - Note Sliding Algorithm

Description: This test covers the reliability and usability of the algorithm that sends the animated notes based on .midi data down a path towards the piano GUI in the application.

Items covered by this test: Note Animation

Requirements addressed by this test: Users can visually see notes in a time-accurate manner that they are supposed to hit.

Environmental needs:

Software Requirements: Piano GUI

Intercase Dependencies:

- NA

Test Procedures: Test will be performed by comparing the timing of each note displaying to a hard-coded timing of each note.

Input Specification: Run song.

Output Specifications: Print the comparison between each note.

Pass/Fail Criteria: If the correct note display timings are printed, test passes.

472# - .midi File Parsing Algorithm

Description: This test covers the reliability of the algorithm used to parse the .midi file data.

Items covered by this test: .midi Parsing Algorithm

Requirements addressed by this test: Users can visually see notes in a time-accurate manner that they are supposed to hit.

Environmental needs:

Software Requirements: .midi File

Intercase Dependencies:

- NA

Test Procedures: Test will be performed by parsing a .midi file, and comparing the parsed data to data from a non-parsed .midi file.

Input Specification: midi File

Output Specifications: Print difference between parsed .midi data and non-parsed .midi data

Pass/Fail Criteria: If there is no difference between the parsed data and the non-parsed data, tests pass.

473# - Song Scoring Algorithm

Description: This test covers the reliability and usability of the algorithm that scores the users' piano playing.

Items covered by this test: Player Playing Grading

Requirements addressed by this test: User receives Artificially intelligent feedback based on how they performed on the piano.

Environmental needs:

Software Requirements: .midi file parser, note sliding algorithm

Intercase Dependencies:

.midi Parsing Algorithm

Test Procedures: User will perform a song, and AI will score the performance.

Input Specification: User Performance

Output Specifications: AI's scoring process and final score.

Pass/Fail Criteria: If the AI scores the song correctly, the test passes.

10 Test Results

#470 MIDI Controller Connection

Date(s) of Execution: March 4, 2022

Staff conducting tests: Ryan Jasiak

Expected Results: Solid MIDI Controller Connection

Actual Results: Solid MIDI Controller Connection

Test Status: PASS

#471 Sliding Note Algorithm

Date(s) of Execution: Feb 12, 2022

Staff conducting tests: Beyza

Expected Results: Notes slither down the path in a time-accurate manner.

Actual Results: Notes slither down the path in a time-accurate manner.

Test Status: PASS

#472 .midi File Parsing Algorithm

Date(s) of Execution: Feb 10, 2022

Staff conducting tests: Ableton Live

Expected Results: .midi file data is parsed to perfection

Actual Results: .midi file data is parsed to perfection

Test Status: PASS

#473 Song Scoring Algorithm

Date(s) of Execution: April 2, 2022

Staff conducting tests: Edward

Expected Results: Song is scored in a sensible and meaningful manner

Actual Results: Song is scored in a sensible and meaningful manner

Test Status: PASS

IV Inspection

11 Items to be Inspected

- Correct use of Camel Casing
- Descriptive Variable and Method Naming
- Ample use of Comments for Non-Descript Sections of Code
- Follows Consistent Spacing and Indentation

12 Inspection Procedures

The inspection process is as follows:

- Each member will pick any random candidate for inspection.
- The candidate will notify the inspector of the code they have contributed to the project.
- The inspector will go over the items to be inspected and alert the candidate of any breaches to the guidelines and mark them down.

There was only one meeting held to discuss and perform the inspection. This was done on April 23, 2022. The results were discussed over an online medium with voice chat and messaging capabilities.

13 Inspection Results

Table 1 - Inspection Form

Inspector: Edward	Candidate: Abel	Date Conducted: 4/23/22	Notes: All the code Abel submitted was inspected for the above items in Section IV.11. After going through the code in great detail, Abel follows all the predetermined guidelines. There were no issues or breaches meaning there were no required resolutions from Abel.
Inspector: Ryan	Candidate: Beyza	Date Conducted: 4/23/22	Notes: Bezya's code was top-notch, 10/10. Her code was inspected using the latest Java tools, along with the information presented in Section IVXI.III. Only issue that sort of presented itself was the keyboard GUI was not easily resizable.
Inspector:	Candidate:	Date	Notes: Edward's code

Beyza	Edward	Conducted: 4/23/22	follows all the coding guidelines from the above section. There were no issues in any regard and passed all inspection quotas.
Inspector: Abel	Candidate: Ryan	Date Conducted: 4/23/22	Notes: Items detailed in section IV.11 in Ryan's code were inspected using the processes detailed in Section IV.12. The code was thoroughly shown to have exceptionally well and descriptive variable and method naming schemes, comments were displayed, and camel casing and consistent indentation and spacing were used. I did not come across any issues that may have arisen as part of the inspection procedures.

V Recommendations and Conclusions

All current items pass the inspection and testing. At this moment, there is no need for any processes or actions to be taken regarding the matter.

VI Project Issues

14 Open Issues

1. Keyboard GUI cannot light up on user input while the song is playing, or it crashes. This is because of JavaFX's limitations in updating the GUI.
2. Scoring Algorithm only provides a score out of 10, rather than in depth feedback based on the users playing
3. Sheet music does not get displayed along with note animations.
4. The cost of MIDI Controllers is predicted to increase by a factor of $O(x^2[\log_2 x^3]^4)$, limiting our user base to those who can afford the increase in price of a MIDI Controller.

15 Waiting Room

1. AI to assess user performance.
2. Connecting to an official database to download songs.
3. We chose to implement the Piano Training Simulator using JavaFX's API. The problem we have been facing with JavaFX is that it is not necessarily meant for frame-by-frame rendering (which would be useful in the determination of the note transitions). Instead, any update to the GUI in JavaFX *must* be sent to the main GUI thread itself. In our case, with so many pieces in the scene being moved and needing to be updated all the time, the main GUI thread gets bottlenecked easily, leading to a crash. Our temporary solution to this issue is to simply limit the frequency of GUI changes to be less than desirable.

16 Ideas for Solutions

Solution for #3 in the above section: A feasible fix to this problem would be to implement the software using a framework that is more suited for frame-by-frame rendering.

17 Project Retrospective

The Piano Training Simulator project was one in which the group reached a common consensus in that it proved to be challenging and frustrating at times, but ultimately was a rewarding experience for the entire team and we each had the remarkable duty of using our strengths to make up for others' weaknesses. We had an urge to create a well thought out and complete implementation of Fall '18 Group 440's (Srinivas Lingutla, Shawn Cody, Tapus Patel, Ji Xiao) documentation, and once we got started we faced many roadblocks, but ultimately it was through the power of teamwork that we were able to succeed. I think some of the biggest challenges we faced were being able to meet up with each other (in-person and online) and coordinating times to communicate and work with each other in which members were mutually free.

VII Glossary

- Piano keyboard: A keyboard only comes with piano's keys that can connect to laptop/desktop via USB 2.0.
- Computer keyboard: The keyboard on computer.
- Sheet music: handwritten or printed form of music notation that uses modern musical symbols to indicate the pitches (melodies), rhythms or chords of a song or instrumental musical piece
- Rhythm: the placement of sounds in time.
- Pitch: how high or low a note is

VIII References / Bibliography

[1] Lingutla, S., Cody, S., Patel, T., & Xiao, J., "Piano Training Simulator", 2018.