Candidate Number: 7629

Jonathan Broster

A windows app gamifies the process of learning piano chords by listening to the player

H446

A-Level Computer Science

A-Level Computer Science

Practical Programming Project

Contents

[Analysis 2](#_Toc145621036)

[Summary 2](#_Toc145621037)

[Further Problem Identification 2](#_Toc145621038)

[Computation 3](#_Toc145621039)

[Decomposition 3](#_Toc145621040)

[Divide and Conquer 3](#_Toc145621041)

[Abstraction 3](#_Toc145621042)

[Data Mining 4](#_Toc145621043)

[Threading 4](#_Toc145621044)

[Conclusion 4](#_Toc145621045)

[Stakeholders 5](#_Toc145621046)

[Research 5](#_Toc145621047)

[Essential features 6](#_Toc145621048)

[Display Chord 6](#_Toc145621049)

[Listen to Notes 7](#_Toc145621050)

[Gather Data 7](#_Toc145621051)

[Learning Algorithm 7](#_Toc145621052)

[Limitations 7](#_Toc145621053)

[Time 7](#_Toc145621054)

[Melody/Tempo 7](#_Toc145621055)

[Requirements 7](#_Toc145621056)

[Testing and Development 7](#_Toc145621057)

[Success Criteria 8](#_Toc145621058)

[Design 9](#_Toc145621059)

[Development and testing 9](#_Toc145621060)

[Evaluation 9](#_Toc145621061)

# Analysis

## Summary

I would like to create a windows app that gives a piano chord then listens for the chord to be played, then automatically gives a new chord.

## Further Problem Identification

Currently there is not a good app to help you practice playing chords on the piano. It is vital to practice playing chords to become a better player, but it can become boring and repetitive. Therefore, I want to create an app to gamify the experience of learning new chords by timing how long it takes you to play the chord. This app could also keep track of what chords you are competent in, and show them less, and what chords you are not competent in playing and show them more. This competency rating can be realised from the time that it takes you to play the chord.

To get a more accurate clock, and so the user knows when they’ve played the correct chord, I need to have the app listening to the user and checking if they’re playing the correct notes. Here is the process of the whole solution, which will loop until a condition is met:

A diagram of a flowchart

Description automatically generated

## Computation

The solution is applicable for computational methods for many reasons. Firstly, the process of listening for a sound and comparing it to a known sound or note can be done quite easily by inputting a section of sound and doing a Fourier transform to convert the wave to a list of frequencies present. I can then discard all the frequencies whose volumes are below a certain threshold, and then convert the remaining frequencies to notes. The computer can automatically display a new chord if the correct notes are detected, whilst keeping track of all the previous chords played and how quickly, to train the user to play the chords much quicker.

### Decomposition

The project can be split into three main parts which should be able to work independently of each other:

* Microphone/Listener/Note Identifier
* UI
* Game – question storage and tracker

By splitting the project in this way, I can ensure that the project can be adapted to meet the needs of any other potential clients other than the stakeholders, so that the code is versatile, modular and works on many devices. I should also be able to employ abstraction in these three areas, so that when developing the UI, I do not need to code the game directly, I only need to interface with it. Furthermore, it will make the project easier to debug because I will be able to more easily identify in which section of the project the bug is in, rather than having to debug the entire code.

The solution involves an algorithm which has some steps:

1. Listen for a sound.
2. Convert the sound from a sine wave to a frequency chart by Fourier transform.
3. Scan the frequencies found to compare against expected frequencies of notes.

This is how the computer will detect chords that the user will play.

### Divide and Conquer

Solving the above decomposed problems together seems technically challenging. To be able to write my solution to the problem efficiently and easily, I will need to conquer each of the decomposed problems separately. I will even divide these components into smaller algorithms and subprograms that seem more manageable on their own.

### Abstraction

Each part of the solution will be abstracted from the others. At the top, there will be the UI, and what the user sees. Then, underneath that there will be the game, which will load new chords and keep track of the user’s competency of the chords. Underneath the game, there will be two components: the file system and loading, because it would be a good idea to store the chords in secondary storage so that the user can save their progress. Also, there will be the note identifier, which takes in frequencies and compares it to notes then outputs any notes that it hears. There will be one more component underneath that which listens and converts the sine wave to frequencies.

Here is a visualisation of that:

A diagram of a computer

Description automatically generated

### Data Mining

The project will implement a simplified version of data mining where I will collect data such as time to find each chord, how many wrong notes were played before the chord was detected, etc. I can then use this data to show the user chords that they find more difficult to play more often, so that they will learn faster.

### Threading

To solve my problem, I will need to utilise multiple threads, as the computer will need to be listening, counting, and checking the previous notes at the same time. My threads will need to pass data to one another so I will implement a lock on some global variables to allow this. One example of where this will be useful is the thread that is listening passes the frequencies to the game, to check that the notes are correct.

### Conclusion

Because of all of the above-mentioned computational methods, the solution is very clearly solvable by computational methods. In fact, the solution can only be solved effectively by computational methods because a human would not be able to identify the chords or provide new ones with enough accuracy.

A picture containing text, font, white, algebra

Description automatically generated

## Stakeholders

Ian Broster

* Piano player.
* Will use a laptop to run the app, on windows.
* Uses an upright piano (with actual strings, not electric, so the frequencies may not be perfect)
* Needs to practice chords, often struggles to find chords quickly, but can work out the chord given time.

Sarah Smith

* Piano player in college, been playing for 1 year, using online lessons to get better.
* Wants to learn piano chords so that she can better play her favourite songs.
* Wants clear, easy to understand chord diagrams.
* Needs a user-friendly interface and colour-blind support.
* Wants the chords to customise so she can practice the ones she needs to.

Mr Johnson

* 45-year-old piano instructor with a wide variety of students from kids to advanced adults.
* Wants to find an app that complements his lessons by offering effective chord practice sessions.
* Needs an app that allows him to customise chord practice sessions for his students, and track their progress, adjusting the difficulty level based on individual needs.
* Values detailed chord explanations and a library of chord variations to challenge his advanced students.

A close-up of black text

Description automatically generated with low confidence

## Research

#### “Yousician”

A screenshot of a music website

Description automatically generated

* An app that helps people learn various instruments by showing the notes and listening to the user play them. Doesn’t focus on chords, but more like reading traditional sheet music.
* Things I like and could incorporate into my app:
  + Plays the chord so that you can learn to hear what it should sound like.
  + Gamify, score system.
  + Piano at bottom of screen showing notes.
* Things I don’t like:
  + Too many tutorial videos.
  + Tailored much more towards beginners.
  + Too much background tune.

#### “Simply Piano”

A screenshot of a video game

Description automatically generated

* A large, versatile piano app that helps the user to learn how to play popular songs. Overly broad range of things you can do on it, not great for more advanced learners.
* Provides songs based on the courses that the user has completed.
* Shows sheet music, and helps you learn all aspects of music from the very beginning.
* Some things I could take from this include:
  + The sheet music to show which notes to play.
  + Videos/cartoon graphics showing which notes to play.
  + The learning curve and complexity scale.
  + The UI is neat and minimal – quite simple and no way to get confused.
* Some things I don’t like about Simply Piano include:
  + Only available for mobile devices
  + You can’t skip around the music or slow down and speed up the tempo.
  + Expensive subscription.
  + No option to turn of backing track.

### Conclusion

Based on this research, I can conclude that I do not want to over-complicate the app, as that is the downfall of many of the larger piano-learning apps. They try to listen to whole songs and rhythm too, and most of the time it doesn’t work very well.

A black text on a white background

Description automatically generated with medium confidence

## Essential features

### Display Chord

The chord name (e.g. Cmin11) should be displayed boldly as the main subject in the screen. The chord name should update on each round of the game when the user moves on to the next chord. This can either be by getting it right or getting it wrong.

### Listen to Notes

After a chord is displayed, the app needs to listen to the user playing notes to deduce if the user is playing the correct notes. This can be done by listening to small amounts of audio regularly through the microphone and doing a FFT to determine the frequencies present, and then checking the frequencies to make sure the notes are correct.

### Gather Data

Whilst the user is playing, the app should gather data such as how long it took the user to get the answer correct and how many tries it took, or incorrect notes.

### Learning Algorithm

The app should then use the data collected to give the chords to the user in a better order, giving some chords more frequently depending on the user’s apparent competence in the chord.

## 

## Limitations

### Time

One of the biggest limitations is time. I have only a couple of months to design and create this app, so some features must be left out. However, in the section labelled “Optional Features”, there are features that could be implemented if I have time, features that would be nice to have but aren’t essential.

### Melody/Tempo

The app will not have capacity to recognise the tempo or rhythm of the user’s playing, just the notes that they play.



## Requirements

The user’s device must have a microphone to detect the notes, as well as support for .exe programs, so it must be a windows device. Without a microphone, the app should “strongly suggest” plugging in a microphone, but still work, moving on with a button click instead. The user must have a piano or keyboard that is in tune with standard tuning.

### Testing and Development

Visual Studio C# Win Forms Application – a useful IDE to develop and test the code. “The Visual Studio IDE is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to enhance the software development process.”[[1]](#footnote-1)

A picture containing text, font, white

Description automatically generated

## Success Criteria



|  |  |  |
| --- | --- | --- |
| Criteria Number | Description | Achieved? |
| 1 | Note Detection |  |
| 1.1 | The program listens and saves the recording in RAM |  |
| 1.2 | The program converts the sine wave to frequencies |  |
| 1.3 | The frequencies are translated into notes |  |
| 1.4 | The notes are saved |  |
| 2 | Difficulty/Revision – Choice of chords |  |
| 2.1 | The program stores lots of chords, and their notes in a large structure. |  |
| 2.2 | The program chooses a chord based on some data collected |  |
| 2.2.1 | Each chord has a “score” which is how well the user knows it |  |
| 2.2.2 | The score is re-calculated each round based on data |  |
| 2.3 | Chords are chosen by score, but never repeated directly (can be ABAB but not AABB) |  |
| 2.4 | The user can select a difficulty |  |
| 2.5 | The user can set favourite chords which have a much higher “score”, so they are played more |  |
| 3 | Game |  |
| 3.1 | The game gives the user a chord |  |
| 3.2 | The game checks any notes currently being played against the chord notes |  |
| 3.3 | If the notes are correct, the game gives the user a new chord |  |
| 3.4 | There is a score that counts how many chords the user has correct, like a streak |  |
| 3.5 | The game shows the user the correct answer after it is played incorrectly |  |
| 3.6 | The user can favourite a chord mid-game |  |
| 3.7 | The user can press “next” to skip a chord |  |
| 4 | UI |  |
| 4.1 | The GUI is clean and simple |  |
| 4.2 | The score is displayed |  |
| 4.3 | The current chord is displayed |  |
| 4.4 | There is indication that the chord is favourited, difficult, etc. |  |
| 4.5 | The background changes to green with ticks when the answer is correct, and red with crosses when the answer is incorrect. |  |
| 4.6 | 4.5 is easy to understand for a colour-blind person |  |
| 4.7 | Next button |  |
| 4.8 | Streak must be displayed |  |
| 4.9 | Some indication of the correct chord visually |  |
| 5 | Catches/robustness |  |
| 5.1 | No microphone – recommends to the user that they should plug in a mic but continues anyway. |  |
| 5.2 | Badly Tuned piano – doesn’t respond to notes that are too far away from the accurate frequency |  |
| 5.3 | All inputs validated |  |

# Design

# Development and testing

# Evaluation

1. https://visualstudio.microsoft.com/ [↑](#footnote-ref-1)