Candidate Number: 7629

Jonathan Broster

A (web) app that controls a Database of suppliers for a company.

H446

A-Level Computer Science

A-Level Computer Science

Practical Programming Project

Contents

[Analysis 1](#_Toc137033106)

[Summary 1](#_Toc137033107)

[Computation 1](#_Toc137033108)

[Stakeholder 1](#_Toc137033109)

[Research 1](#_Toc137033110)

[Essential features 1](#_Toc137033111)

[1](#_Toc137033112)

[Limitations 1](#_Toc137033113)

[Requirements 1](#_Toc137033114)

[Success Criteria 2](#_Toc137033115)

[Design 2](#_Toc137033116)

[Development and testing 2](#_Toc137033117)

[Evaluation 2](#_Toc137033118)

# 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.

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.

#### “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. Very broad range of things you can do on it, but not great for more advanced learners.

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 in order 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.

### Microphone

Another limitation is the hardware. Not all laptops and devices contain good microphones, which could make the notes difficult to pinpoint amongst background noise. This is down to the user’s hardware and largely, it can’t be helped. However, I could implement a decibel-based system for determining notes so that the threshold for the notes to be counted as notes picks them up easier. This would involve using a logarithmic scale for amplitude.



## Requirements

A picture containing text, font, white

Description automatically generated

## Success Criteria

TODO List



|  |  |  |
| --- | --- | --- |
| Criteria Number | Description | Achieved? |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Design

# Development and testing

# Evaluation