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

## 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

A picture containing text, font, white, algebra

Description automatically generated

## Stakeholder

Ian Broster – piano player

A close-up of black text

Description automatically generated with low confidence

## Research

A black text on a white background

Description automatically generated with medium confidence

## Essential features

## 

## Limitations

TIME



## Requirements

A picture containing text, font, white

Description automatically generated

## Success Criteria

TODO List



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

# Design

# Development and testing

# Evaluation