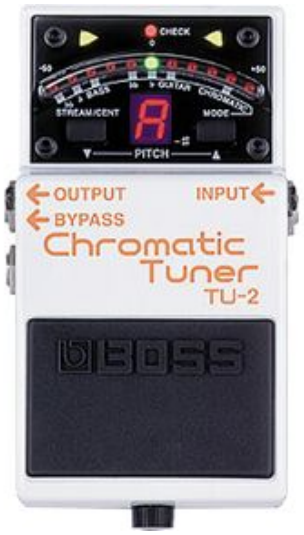
CRAIG SPENCE - 08497818

281.429 - APPLIED MULTIMEDIA SIGNAL PROCESSING

AUDIO EFFECTS PROJECT - PROJECT PLAN

Chromatic Tuner & Auto-transcription

PROJECT CONCEPT:



A chromatic tuner is a device that many musicians (particularly guitarists/bassists) use every day. They come in many physical shapes and forms for example:

- Foot-pedal
- Rack-mounted
- Hand-held
- On-board a guitar

And, of course, there are web-based/mobile-app tuners available of varying degrees of complexity. At the present these are roughly divided into **two** categories:

1. Those that play a reference pitch to which the user can adjust their instrument to.
2. Those that use **Java** or Flash to access the users microphone and respond to the actual pitch of an instrument.

As the web develops and evolves as an application platform, most notably with the advent of **HTML5**, new **APIs** and features are becoming available from within the browser that haven't been accessible before. Included in the set of new features are capabilities to access a user's microphone from the browser and manipulate the audio data received.

A second group of tools that are also frequently used by guitarists and bassists are online tablature and chord databases. These typically contain by-ear transcriptions of a wide variety of songs, in one of two standard forms, tablature (figure 1) and basic chord structure, normally with song lyrics (figure 2).

TABLATURE:

```
G|-----.---------------------------|----------------------2-3-2-------|
D|-----5---5---5-3-.---------------|-.-------.-------.---------3-0---|
A|-----------------5---5---5---3-5-|-6---6---6---6-5-3---------------|
E|-3-------------------------------|---------------------------------|
   1 + + + 2 + + + 3 + + + 4 + + +   1 + + + 2 + + + 3 + + + 4 + + +
```

Figure 1 - An example of bass guitar tablature. The four lines represent the four strings, and the numbers represent fretting positions. This example also includes a basic representation of articulation and timing.

Tablature is essentially a top down depiction of the neck of a guitar or bass, with the lines representing the strings. The letters to the left indicate the appropriate tuning. The numbers represent fret position, with '0' being an open string. Some examples of tablature include rudimentary timing notations as well as articulation (such as slides, hammer-on/pull-offs, bends). Tablature is generally considered to be easier to read quickly than standard musical notation, which has been a contributing factor to its popularity.

CHORDS & LYRICS:

```
                F                   G                C
Now who'd have thought, that after all, something as simple as rock 'n'
C/B                Am
roll would save us all?
               F7                  G
Now who'd have thought, that after all, it was rock 'n' roll!
```

Figure 2 - An example of a chord chart for a song. Chord changes occur on the word underneath the chord letter. There is the assumption of some pre-existing knowledge of the song and its melody.

A chord chart is another level of abstraction away from standard notation. They provide a general overview of the structure of a song, perhaps to be played with an acoustic guitar. At the most basic representation, a capital letter indicates a major chord from the twelve-note (A, A#, B, C, C#, D, D#, E, F, F#, G, G#) chromatic scale. However, These chord names are often embellished:

- An **'m'** following the letter name indicates a minor chord
- A **'/'** followed by a different letter indicates an altered root note (the lowest note of the chord)
- Additional numbers following the chord names indicate that an extra note (such as a 7th interval - **'7'**) should be added.

PROJECT SCOPE:

KEY COMPONENTS:

This project has **three** parts:

1. Investigate extracting accurate frequency data from such an audio stream with the aim of creating an interactive chromatic tuner that responds to audio input from the user's microphone. Ideally this will be achieved solely using **HTML5 APIs** with **JavaScript** on the client, but if this is not yet technologically feasible, it will be implemented as a **Java** applet.

2. The frequency-based pitch recognition from the tuner will be combined with time-domain signal analysis to perform automatic transcription of simple audio files (one guitar/bass instrument), in order to generate rudimentary tablature.

3. The complexity of automatic note-for-note transcription of complex audio files is beyond the scope of this project. Instead, the third part of the project will look at the overall structure of a multi-instrument audio file and attempt to identify the fundamental chord arrangement.

DELIVERABLES:

The main deliverable for this project will be a web-based application (written in **CoffeeScript**, **HTML** and **CSS**) capable of receiving some user input (either in real-time from a microphone or an existing audio file) and processing that input, generating one of **three** outputs:

1. For real-time microphone input, the application will output the closest pitch note from the twelve-note chromatic scale - measured in hundredths of a semi-tone (cents) - and a direction in which the pitch must change to become in tune. This will be based of the common standard that an **'A'** is **440hz**. There will also be the assumption of a modern equal-tempered interval system rather than the natural Pythagorean system.

2. For existing audio files, the application will be able to attempt to generate tablature output from the input. It will assume that the input is from a guitar instrument. It should recognise notes outside of the usual range of a standard-tuned guitar and recommend that the tab be displayed with an alternative tuning. The user should also be able to specify the tuning. The output should be as playable as possible, by minimising the jumps between positions where there are multiple options. It should be able to determine individual note timings to semi-quaver accuracy, and detect common time signatures.

3. For existing audio files, the application will also be able to attempt to generate a chord chart describing the overall tonal structure. There will be no assumption of the instruments that are played in the file. For a particular time-subdivision, dominant pitches will be determined and a chord name will be associated with that time period.

Another key deliverable will be code documentation, along with appropriate unit testing.

MAJOR CHALLENGES:

This project has several main challenges that will need to be overcome for a successful outcome:

- Getting audio data into the browser in a form that can be programmatically manipulated
- Performing frequency domain decomposition (implementing the FFT algorithm) and determining what pitches are relevant and which are harmonics
- Assessing the pitch of microphone input fast enough for the output to be usable in real-time like a hardware tuner is
- Determining tempo and beats using an existing technique such as autocorrelation.
- Time decomposition and determining where changes happen within a bar of music.
- Determining chord names from note groupings.
- Generating readable, playable outputs.

RISKS:

MAJOR RISKS:

This project has some risks that may get in the way of a successful outcome:

- It may not be possible to access audio data from the microphone using pure **CoffeeScript**, as the **APIs** may not be mature enough to provide the necessary functionality.
- The project scope may be too big to achieve within the time available, depending on how difficult it is to overcome the main challenges.
- Loss of data/time from not backing up
- Software may not perform as it is intended to

RISK MANAGEMENT:

The above risks will need to be managed to ensure that they do not prevent the successful completion of the project:

- The first major risk will be managed by making a judgement at the end of the first week as to whether the functionality is available to make the application work in pure **CoffeeScript**. If it is determined that it is not possible, the application will be written in Java and run in the browser as an applet.

- The second major risk will be by making an assessment two-thirds of the way through the project, to decide whether to minimise the scope of the project to two of the three desired outcomes and focus on perfecting those components. While this is obviously not an ideal outcome, for the purpose of this project it would be better to have two things working well than three not working at all.

- The **GitHub** website will be used as a software repository and issue tracking system, which involves using for version control meaning there will always be multiple copies of any work done.

- **QUnit** (for **CoffeeScript**) or **JUnit** (for **Java**) testing will be done as much as is practical with the time constraints. An emphasis will be placed on testing project-critical code rather than details such as user-interaction etc.

TIME MANAGEMENT & MILESTONES:

With the project completion date set as **Monday 10th September, 2012**, there are six and a half weeks available to complete it (from **Wednesday 1st August, 2012**). Allowing for the last few days to be for documentation, that gives **six** weeks of development time, or **two** weeks for each main component.

The first two weeks will be dedicated to creating the real-time chromatic tuner:

- Week **one** will focus on proving that the application is possible with currently available **APIs**, and making a decision if the project needs to change to **Java**
- Week **two** will focus on developing strong frequency analysis functionality in either **CoffeeScript** or **Java**, depending on the outcome of week **one**. Having a well-tested foundation will make the other goals more achievable.

The second two weeks will focus on the time and frequency analysis of simple audio files to generate accurate tablature:

- Week **three** will focus on building on the foundations from the first two weeks, and adding code to decompose an audio signal in the time domain.
- Week **four** will focus on turning the time & frequency data into a form that is understandable for a user. At this point, a decision will be made about how feasible it will be to achieve the third component to an acceptable level within the time allowed.

Depending on the project status at the end of week **four**, the final two weeks will involve either:

- Catching up with the requirements for the first two components and improving their results.

or:

- Generalising the time and frequency analysis to assess the changes in chords in audio files through time.

IMPORTANT TERMINOLOGY:

API: - Application Programmable Interface: a specification used as a way for software components to interact with one another.

COFFEESCRIPT: - A programming language that compiles into JavaScript, but has the advantage of being much nicer to read and write. Also protects the programmer from some common pitfalls of JavaScript.

CSS: - Cascading Style Sheet: documents that describe the styling of a web page by specifying properties that the web browser applies to the specified elements.

GITHUB: - Online software development community and source-code repository, featuring version control with Git and issue tracking.

HTML/5: - HyperText Mark-up Language: the programming language used to describe web pages. HTML5 has come to encompass the next generation of web functionality, including things like location tracking, touch interface controls etc.

JAVA: - General purpose programming language that can do pretty much anything. Can be embedded into web pages as **'applets'**, but this requires the user to use a plug-in.

JAVASCRIPT: - The dominant programming language for client-side code on the Internet. JavaScript has some significant pitfalls which can lead to inelegant (and sometimes dangerous) code, but as they are well

known, they can be avoided. Javascript is currently going through a renaissance as a server-side language.

JUNIT: - Industry standard unit and regression testing framework for Java.

QUNIT: - JavaScript implementation of JUnit to allow for easier unit and regression testing of JavaScript code.