CS Capstone Final Report

June 12, 2018

Music Theory Application

Lukas Hein _____ _____
Signature *Date*

Prepared by

Group 45
Team TriTone

Aidan O'Malley _____ _____
Signature *Date*

Christopher Hebert _____ _____
Signature *Date*

Abstract

This document is a compilation of all of the work of Group 45's (Team Tritone) experience in Capstone working on the Perfect Fifth music theory app. It includes the original requirements, design, and technical review documents that were signed and approved by their client, Lukas Hein. It includes weekly blog posts and project documentation, as well as a retrospective from each team member.

CONTENTS

1 Introduction To Project

Perfect Fifth is an iPhone and Android app aimed at teaching beginning composers the fundamentals of music theory in such a way that they can immediately apply the concepts toward creating their own music. The app presents our client's, Lukas Hein, vision of music theory, which he calls the schedule of tonal gravity. It provides a simple foundation for choosing which chords resolve to each other using a tool known as the circle of fifths.

The main page of Perfect Fifth is an interactive circle of fifths. The app also provides reference pages for learning the fundamentals: including information about notes, intervals, keys, chords, and how to use the circle of fifths and tonal gravity to compose. The app gives the musician a composition page that allows her to try composing phrases digitally, and have the app analyze the phrase according to tonal gravity theory.

As a team, Aidan O'Malley, Chris Hebert, and Lukas Hein took on the task of planning and designing the Perfect Fifth application. Aidan took on the responsibility of designing and implementing the circle of fifths page, working closely with Lukas. Chris designed and programmed the composition page. Lukas took an active role in the app's development by writing the reference pages, condensing his music theory knowledge into bite-sized chunks.

2 Requirements Document

Oregon State University
College of Engineering

CS Capstone Original Requirements Document

June 12, 2018

Music Theory Application

Lukas Hein _____ Signature _____ Date

Prepared by

Group 45

Team TriTone

Aidan O'Malley _____ Signature _____ Date

Christopher Hebert _____ Signature _____ Date

_____ Signature _____ Date

Abstract

The requirements for the Interactive Music Theory Application are described. The scope and purpose of the project, the intended hardware platforms are laid out. Specific user stories describing the purpose and function of each app page are presented. The application is divided into pages, and each pages interactions are detailed including visual layout, layout of touch buttons, and audio playback. A timeline of implementation is given.

3 INTRODUCTION

3.1 Purpose

The purpose of this piece of software is to provide an educational musical resource to people of all ages and experience levels. Specifically, the application focuses on the functionality of harmonies in modern music and how to apply them. The application intends to address many problems with how music is taught currently such as the excessive sources of information, unnecessarily difficult information, and necessary presence of a teacher to teach music theory concepts to name a few of these problems.

The intended audience of this piece of software is wide. Any person with interest in learning music theory would have interest in this application along with people looking to write their own music and improve their composition skills. The presentation of the concepts of the schedule of tonal gravity and the function of the interval of fifths will be easy enough for children to be able to utilize the application, however, some of the more complex concepts will be focused more on older and more experienced audiences. Because of the ease of use and the wide-reaching concepts touched on, the audience for this application will be anyone interested in learning more about music theory and composing modern music.

3.2 Scope

The software product to be produced by Team TriTone will be Tonal Gravity.

The product will focus on teaching musical concepts in an interactive manner. Specifically, the concepts taught will center upon the circle of fifths and will use the graphic to help users understand the concepts with a familiar basis. The concepts that will build upon the circle of fifths are the schedule of tonal gravity, parallel keys, relative keys, tritone substitutions, secondary dominants, diatonic substitutions, and diminished substitutions.

The application of the product will be as a mobile application on the Apple App Store and the Google Play Store. The application strives to improve the quality of musicians around the world by spreading easily understood and very general compositional techniques.

3.3 Overview

The most basic and core function of this application will be the circle of fifths which displays the corresponding notes within the selected key. From here the user can select options that would allow them to view parallel and relative keys, or select different keys.

To supplement this core functionality we will creating separate pages one of which will describe how and why the circle works along with some general rules of the schedule of tonal gravity and descriptions of vocabulary used within the tool. Another page will allow the user to input compositions which the application will then analyze and output information with regards to the users composition. For instance it might inform the user if they followed the rules of tonal gravity or possibly recommend how to improve the composition. To further supplement the users comprehension of the core tool, we may also add quizzes or interactive media that cements the users understanding of musical theory.

4 Overall Description

4.1 Product Perspective

The app is self-contained, and does not interact with any other applications or protocols.

It will be released for the iPhone, iPad, and Android tablets and phones. It needs to be developed for touch interfaces and limited screen space. Users of the apps are expected to use thumb and index fingers. It should operate under a variety of resolutions. It should be able to access enough flash memory to save and load compositions. It should be able to access the audio playback operations in order to play user compositions.

4.2 Product Functions

The main function for this product is to provide a simple way for new musical students to understand the basics of musical theory. This will be accomplished by using an interactive circle that describes the relationship between notes. Along with the supplementary reference pages the users should be able to use this tool to gain some insight on bring notes together in a logical manner. The secondary function of this product is to be used as a tool for composers. This will be accomplished by providing, in a sense, a map of the different possible routes for traversing the musical terrain.

4.3 User Characteristics

There will be two main use cases for this tool. The first will be for musical students learning an instrument or just curious about musical theory. For this use case the tool should be able to describe the basics of how to derive logical and objectively pleasant music. This will give the newly curious individual some insight on why some groups of notes sound better together than others.

The other use case for this tool will be for composers attempting to better understand how to adapt chord progressions to their will. For this use case the tool should be able to aid composers in discovering the options they have for their choice in progression. This could be in the form of attempting to link keys together by using the tool as a mapping device between destinations. This could also just as well help them to discover their creativity in exploring new routes of key progression.

There wont be too many expectations on the users of the this tool as it will be largely educational and not require much for knowledge of musical theory to understand. Though it can be used as a tool the main function of the application is still to help educate people on music theory and therefore should be able to answer many of the common question for the novice musical student. The only expectation I could see for the recurring user, is an interest in musical education.

4.4 Constraints

The constraints on this project will mainly be time and scope. Research still needs to be done to determine all the details of what features should not be approached or covered by the application as well as which features in addition to the core application we will have time to implement. There will also be many obstacles in implementing the application including; determining the workload of each group member, getting the application to work on both android and apple platforms, verifying the usability of our product, and releasing the application the respective stores. The related topics that could be included in the application or the application built off of this tool are very numerous so a big constraint for this project will be determining how we design the framework to allow for adaptability.

4.5 Assumptions and Dependencies

This SRS assumes that the app can be written for a virtual machine which provides: a touch screen interface, audio playback, visual display, timers, and permanent memory. The code written for the virtual machine should be automatically translated to code that executes on the various devices. If this is not possible, the development of this app will be limited to the iPhone and iPad devices only.

5 Specific Requirements

5.1 Interactive Circle Of Fifths: 00

- Spinnable by user
- Correct notes are on the outside of circle and transition correctly when spun
- Correct numbering above notes within selected key

5.2 Circle of Fifths sidebar: 01

- Draggable by user, when dragged, circle of fifths should react by spinning
- Sidebar should reflect the correct chord quality based on the current key

5.3 Parallel and Relative Keys within Circle of Fifths: 02

- Possible to add a keys parallel and relative keys to the circle of fifths iteratively
- Visually easy to understand the flow between keys
- Color coded
- Proper numbering of chords in parallel and relative keys
- Spacing is correct and doesnt look cluttered

5.4 Composition Page: 03

- User has ability to create a few measures of chord progression
- User has an easy way of inputting chords whether it be with the circle of fifths or an altered keyboard
- Application has the ability to properly analyze a composition and determine if a user followed the rules of tonal gravity
- User has ability to iteratively add parallel and relative keys to the composition analysis

5.5 Reference and Tutorial Pages: 04

Reference pages are pages of text and non-interactive visuals describing the various music theory concepts.

- Schedule of Tonal Gravity reference page
- Parallel and Relative Keys reference page
- Secondary Dominants reference page
- Tritone substitution reference page
- Diatonic substitution reference page
- Diminished substitution reference page
- Frequency and Interval reference page
- Chord reference page
- Vocabulary page

6 Development Flow

Rough Draft Note: Make a Gantt Chart from the following data:

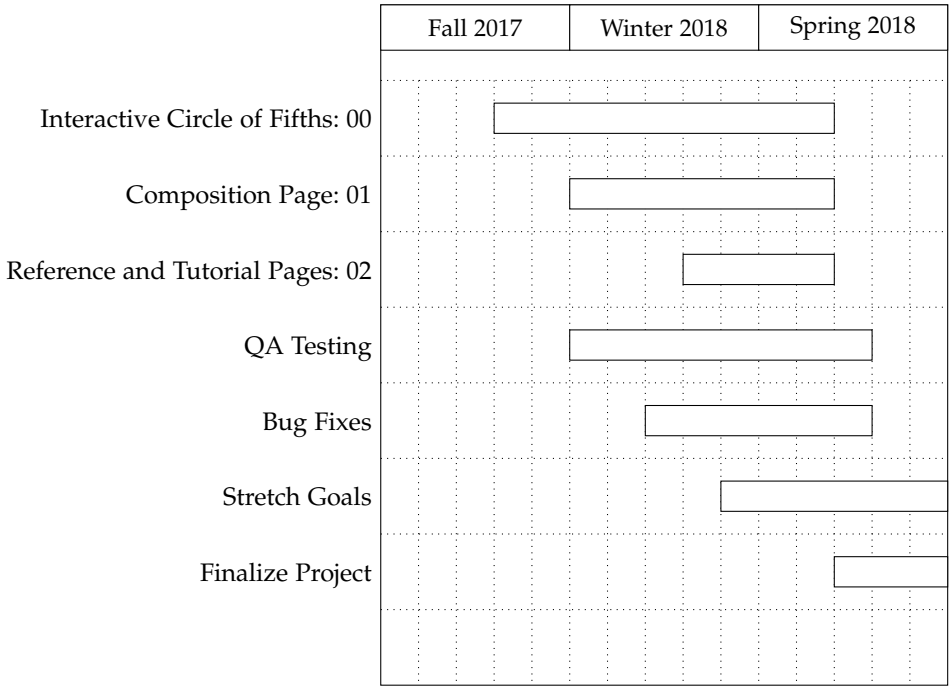
- 11/17 - 1/18 Interactive Circle of Fifths: 00
- 1/18 - 2/18 Circle of Fifths Sidebar: 01; Depends on 00
- 1/18 - 2/18 Parallel and Relative Keys: 02; Depends on 00
- 11/17 - 2/18 Composition Page: 03
- 11/17 - 2/18 Reference and Tutorial Pages: 04

6.1 Updates to Original Document

The following approved and/or requested changes were made to the original requirements document:

- The app no longer should be able to access enough flash memory to save and load compositions.
- The app no longer should be able to access the audio playback operations in order to play user compositions.
- The user can see the correct notes inside of the circle, instead of outside of the circle.
- The user can change root note of circle by rotating the circle or using the sidebar
- The app no longer requires that the user can see that the sidebar reflects the correct chord quality based on the current key.
- The app no longer requires that the user has the ability to iteratively add parallel and relative keys to the composition analysis
- The app no longer requires the reference pages: Parallel and Relative Keys, Secondary Dominants, Tritone substitution, Diatonic substitution, Diminished substitution, Frequency and Interval, Vocabulary.
- The app now requires the reference pages: Perfect Fifth, Enharmonic Equivalents.

7 Final Development Flow



8 Design Document

Oregon State University
College of Engineering

CS Capstone Final Report

June 12, 2018

Music Theory Application

Lukas Hein _____ _____
Signature Date

Prepared by

Group 45

Team TriTone

Aidan O'Malley _____ _____
Signature Date

Christopher Hebert _____ _____
Signature Date

_____ _____
Signature Date

Abstract

The purpose of this document is to outline the system architecture and design decisions made for the Music Theory Application. The scope and purpose of the application is summarized. The system architecture and implementation language are specified. The pages of the application are laid out and their implementations are described.

CONTENTS

9 INTRODUCTION

9.1 Scope

The piece of software discussed in the following document is intended to be used as an educational resource for musicians of all experience levels. Our goal is to create an application that will enhance the quality of musicians around the world by teaching concepts of functional harmony.

9.2 Purpose

The purpose of the following software design document is to outline the design of the application at a fairly low level. All functionality of the application that is necessary in achieving our teams goals for a completed product are contained in this document.

9.3 Intended Audience

This document is intended for capstone instructors and teaching assistants as well as for our group and clients reference for the upcoming terms. Anyone interested in the development process of the application will also find the following document useful. However, this document will mainly be used to define a set of design goals to be used in comparison with our final product.

9.4 Conformance

This document conforms with the requirements outlined by the client, Lukas Hein, during the many meetings conducted between the client and the team over the course of fall term. Any changes to design will be reflected in revisions to this document.

9.5 Definitions and Acronyms

- **COF**: Meaning Circle of Fifths, the main aspect of our application. It is the circle of the 12 tones of western music with the intervals of four going clockwise around and the intervals of five going counter-clockwise.
- **Sidebar**: The vertical list of the 12 notes in western music that will be placed adjacent to the COF on the applications main page.
- **Composition Page**: The page where a user will be able to create their own composition and test their ability to follow the rules of tonal gravity.
- **Schedule of Tonal Gravity**: The natural direction that a specific note wants to travel when in a specific key.
- **Musical Golden Mean**: BEADGCF. The tonal gravity of the key of C, from left to right shows the direction that each chord in the key of C wants to travel.
- **Tonal Gravity Rules**: The three rules outlined by Lukas for a keys tonal gravity. 1.) Do not go up consecutive steps of the keys tonal gravity. 2.) Do not go down the schedule of tonal gravity by more than one step. 3.) No limit on steps one can skip when travelling up the schedule of tonal gravity.
- **Key Signature**: The notes that make up a specific key.
- **Parallel Key**: The key that has the same root note as our initial key, however, is in the opposite mode. For example, C majors relative key is C minor and visa versa.

- **Relative Key**: The key that contains all of the same notes as our initial key, however, the root note, chord qualities and schedule of tonal gravity differ [10]. For example, C majors relative key is A minor.
- **Interval**: An interval is the distance between two pitches, usually measured as a number of steps on a scale [11].
- **Chord Quality**: The name given to a chord based on the intervals between the root and the other notes that make up the chord [12].

10 System Architecture

10.1 Design

Since our client requested that our project be a mobile application we needed to decide on some tool for creating it. During some of the initial meetings with our client it was clear that we would want the application to run on both android and apple mobile devices and we did not want the application to be web based initially. So moving forward from here we wanted to chose a framework that would make it easy for us to make a product for both platforms. After considering our options for the system architecture we chose to use a framework called React Native. Developing in this component based Javascript framework is much like coding in an object oriented framework because it treats each component much like an object. This allows us to easily select and implement specific components required to create the interface users will be able to interact with our application through.

In this framework we can structure a hierarchy of components where a page can be the container component such as the Circle of Fifths page. This component will then be made up of smaller reusable components like the actual COF, the Sidebar, and the buttons. These inner components can contain components within them as well and our data will flow logically down this hierarchy to the child components. In short, the application will start at our home page of the Circle of Fifths page which is its own component. There will also be container components for the Composition page and the References pages. These will all have unidirectional data flow and will contain similar components that will be reused between them. For example, the COF component will be shared on the main page and the composition page. This structuring provides a very logical and readable codebase and will make our teams lives easier as we progress.

10.2 Rationale

Of the framework options we considered for programming our application in, we decided on React since the framework allows us to implement our application for both Apple and Android devices simultaneously and is very developer friendly. In addition, the power and simplicity of this specific framework would allow us to quickly become familiar with the necessary components needed to design and implement our application.

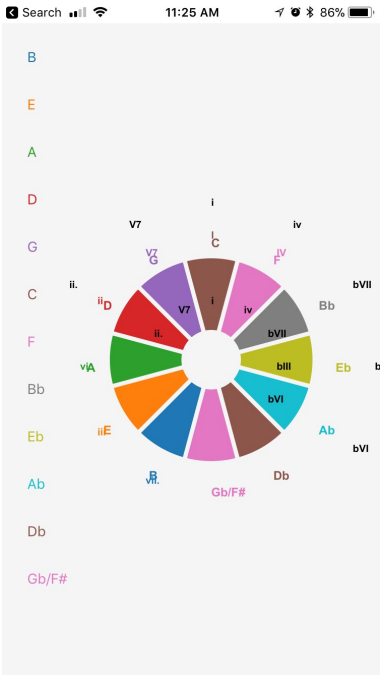
11 Component Design

11.1 Circle Of Fifths Page

The Circle of Fifths page is the main page of our application and will demonstrate most of the concepts that we attempt to educate the user on. It consists of the COF, the sidebar, buttons to add or remove parallel and relative keys, and buttons to travel to reference and composition pages.

11.1.1 COF

11.1.1.1 GUI Design: From a design viewpoint, the graphical user interface for the main circle will consist of a pie chart that has all of the 12 notes with placement corresponding to the 12 wedges of the pie chart. The keys will be represented by the keys chord qualities. The chart will contain concentric rings to split up the relative key, current key and parallel key in that order. The colors of the pie chart will be shades of the rainbow along with five filler shades as a rainbow only has seven colors and we need 12. The colors of text within the circle will contrast that specific wedges color. Below is an image giving some idea of the UI for the circle with some aspects left out.



11.1.1.2 Implementation Design: From an implementation viewpoint, using a vector based graphic that is drawn within the application is the implementation we have decided on for the Circle of Fifths. There are many positives in using a graphic that is actually created within the application's source code. he main positive being we have more customizability over the animations and interactions with the circle. Since it is vector based, there will be no issues with resizing the graphic if necessary. There are also a lot of different options for creating vector based graphics in React Native. We will be creating the COF using the third party javascript libraries ART and d3 such as what is done in the article written by David Vacca to create a pie chart [1]. We will use Reacts built in Text component as well as the ART Text components to create the notes and chord qualities that are involved in the circle and will use basic circle geometry to place these components [5] [9]. We will use Reacts PanResponder and Transform properties upon the circle to rotate the circle when the user gestures a spin [7] [2]. With these libraries and React Native components we will be able to achieve all of what the client has deemed necessary for the COF.

11.1.1.3 Rationale: The rationale behind the design specified is to provide the user with a functioning and interactive circle of fifths that is also aesthitically pleasing and is relatively easy to develop. Since the circle is used for most of the concepts being taught and since it is where most of the interactive learning will be implemented, the COF must be included in the application.

11.1.2 Sidebar

The sidebar is a critical piece of our application as it displays the schedule of tonal gravity for the current key to the user. It consists of the 12 notes in western music in a vertical list.

11.1.2.1 GUI design: From a design viewpoint, the UI design for the sidebar is a vertical list of the 12 notes of western music. The colors of the notes will correspond to the colors of the notes on the actual COF. This is necessary to provide consistency across the application. The notes will be separated far enough apart for the average user to be able to select a specific note easily. The notes of the current key will be highlighted. Below is an image giving some idea of this design.

11.1.2.2 Implementation Design: From an implementation viewpoint, the sidebar on our main Circle of Fifths page will combine both touching and dragging gestures to manipulate the contents of the COF. Touching a specific note will bring that note to the top of the circle and reflect that notes key. Dragging along the length of the bar will bring whichever note the users finger is currently over to the top of the circle and that notes major key will be reflected. The main positive of this implementation is that users will be able to have options for altering the circle's contents and can end up focusing on one specific implementation based on personal preference. If one user's fingers are too big to click the notes, they have the option to drag the bar. If one user likes the dragging animation more than clicking then they can drag. If a user likes to see their changes reflected on the circle immediately they can click a note. Users need to be able to be comfortable with our app and giving options on how to use the app would be a big help in achieving comfort. The implementation will be accomplished using Reacts Text component to place the notes on the screen [5]. We will then use Reacts PanResponder to implement responses to the users touching and dragging gestures; the onPanResponderEnd function will signal a touch or the end of a drag and the onPanResponderMove function will signal a drag [7]. This implementation will allow us to meet all of the goals we have outlined for this piece.

11.1.2.3 Rationale: The sidebar is necessary for a completed application to help users understand the schedule of tonal gravity and to give users another way of manipulating the COF contents. The implementation specified is necessary to give the user a few options of utilizing the sidebar to their preference.

11.1.3 Buttons

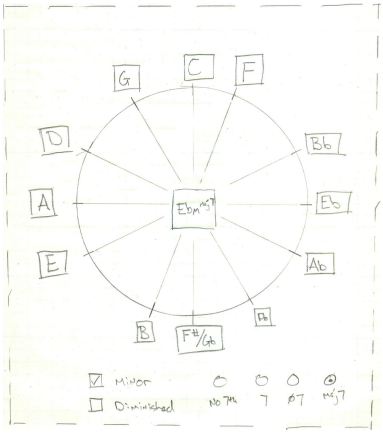
There will need to be multiple buttons on the Circle of Fifths page for adding parallel and relative keys as well as for navigating to different pages within the application.

11.1.3.1 Implementation Design: Buttons within the Circle of Fifths page will be implemented with the React Native Button component [8]. The Button component comes with an onClick property which will be used to tell the application what should happen at this point. If we are adding/removing keys to the COF we will just use the onClick property to call a function which will show/hide the specific keys section on the page. If we are changing screens we will use the React Native Navigator component to tell the application to navigate to the specific View we are navigating to [3].

11.1.3.2 Rationale: The rationale behind the design specified above is that we need buttons to be able to let the user interact with different components of our application and to be able to change pages. There isnt another way of implementing this functionality without the design specified for the buttons above.

11.2 Composition Page

11.2.1 Chord Entry



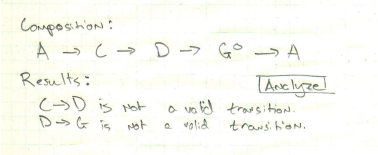
11.2.1.1 GUI Design: The circle of fifths style chord entry uses a circle of fifths for tone entry, as well as two groups of radio buttons for chord quality and 7th interval quality. The user will be able to touch any of the notes on the circle of fifths to determine the root tone of the chord. The chord quality radio buttons are major, minor, or diminished. The 7th interval quality radio buttons are none (n o 7th), 7th, major 7th, or diminished 7th. The chord as it is entered will be displayed in the center.

11.2.1.2 Implementation Design: A React Image Component will be used to implement the circle. A Text Component will be used to implement the display for the chord being entered. This will be updated when it receives events that come from the radio buttons. The note buttons will be implemented using Button Components which set the state of the entered chord. The other groups of radio buttons will be implemented using HTML Radio Input tags, which will also set the state of the entered chord.

11.2.1.3 Rationale: The COF method of chord entry has very high cohesion with the another component of the app: the Circle of Fifths page. This would allow users to develop muscle and spatial memory of the COF as they practice composing.

Implementing the COF style chord entry solves the problem simply and effectively, has the added benefit of drilling muscle memory for users, and has high cohesion with other parts of the app.

11.2.2 Analysis



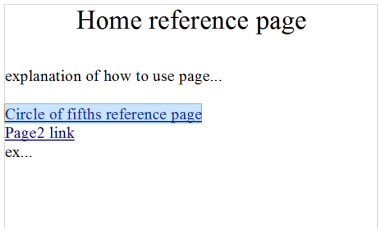
11.2.2.1 GUI Design: From this viewpoint, the user can push a button that will analyze or re-analyze the entire composition. This element has 3 sub-elements: the current composition, the results, and the analyze button. The current composition is made up of a Composition React Text Component, followed by another Text Component that contains the composition that has been entered so far. The results is made up of a Results: Text Component, followed by a series of Text Components representing the results of the last analysis that was run. Finally, the Analyze Button Component will re-analyze the composition when the user presses it.

11.2.2.2 Implementation Design: In order to analyze the composition, the chords, their qualities, and their 7th qualities all need to be stored. These can be stored as a record of note name, quality, and 7th quality. The composition then can be stored as an ordered list of chords. The Composition Text Component can be updated to reflect this ordered list, every time a new chord is added.

When the analyze button is pressed, a function will be mapped over each pair of chords in the ordered list which will determine if the transition from one chord to the next is acceptable according to the rules of tonal gravity. The results will be collected. A result should be a record of two chords, and the number of the rule that was broken.

11.2.2.3 Rationale: A single analyze button was chosen since it is easy for the user to understand, without forcing analysis on the user. This gives the user flexibility to ignore analysis if they feel like breaking the rules of tonal gravity.

11.3 Reference Pages



11.3.1 Implementation Design

The implementation for these reference pages will be created using the React Native text component which the component for the reference page will contain. The container component will utilize the React native (tabbed pages) component to contain the list of pages along with the pages themselves to easily navigate through. This component will allow us to switch between different use cases of the application by listening for an onClick event, much like switching

between tabs on a browser. Once selected it will switch the view from the tab previously selected (either the circle of fifths tab or the composition/ comprehension tab), to the home page of the reference pages. Here it will then display a list of links to the actual pages that describe and explain the different topics used within the application. The main page will also contain a brief description of what can be found on each page and how to go about finding the information the user may be looking for.

As a stretch goal we have also considered linking to these reference pages directly from the aspects of our application that might be explained by a specific reference page. For instance when selected in a certain manner, the circle of fifths may display a pop-up dialogue box that gives a link to the circle of fifths reference page. Another option would be allowing the pop-up itself to display the specific information a user might find helpful based on what they were clicking.

11.3.2 Rationale

Of the other options we considered, we decided that using reference pages would be the cleanest looking option. Although this choice might make it slightly difficult to find the description of the topic the user may need assistance with, we decided it would still be the best option as it would allow us to provide any amount of information regarding a specific topic. This would allow us to create more complete topic explanations and descriptions as we would not need to worry about providing too much or too little data that might confuse the user when compared to placing the references within the alternative options we considered.

REFERENCES

[1] *Animated Charts in React Native using D3 and ART*
`https://medium.com/the-react-native-log/animated-charts-in-react-native-using-d3-and-art-21cd9ccf6c58`

[2] *Transforms*
`https://facebook.github.io/react-native/docs/transforms.html`

[3] *Navigators*
`https://facebook.github.io/react-native/docs/navigators.html`

[4] *Images*
`https://facebook.github.io/react-native/docs/image.html`

[5] *Text*
`https://facebook.github.io/react-native/docs/text.html`

[6] *View*
`https://facebook.github.io/react-native/docs/view.html`

[7] *PanResponder*
`https://facebook.github.io/react-native/docs/panresponder.html`

[8] *PanResponder*
`https://facebook.github.io/react-native/docs/button.html`

[9] *ART.Text*
`http://sebmarkbage.github.io/art/docs/Shapes/ART.Text.html`

[10] *Key Signatures*
`http://openmusictheory.com/keySignatures.html`

[11] *Intervals*
`http://openmusictheory.com/intervals.html`

[12] *Triads*
`http://openmusictheory.com/triads.html`

11.4 Updates to Original Document

The following approved and/or requested changes were made to the original design document:

- Circle of Fifths
  - Circle design changed to be one circle with concentric lines to separate the different keys.
  - The placement of the text for the current key, parallel key, and relative key changed to all be within the circle.
  - Sidebar was moved to the left side and includes more than just the common 12 keys in western music.
  - Instead of buttons to add/hide the parallel and relative keys, we instead used dynamic text that would change based on the action the user would perform if they clicked the text.
- Composition
  - Chord entry became a set of radio buttons instead of using the circle of fifths chord entry.
- Reference Pages
  - This implementation changed to using a menu to change between pages rather than a table of contents page.
  - We also changed the implementation to use a swiper so that a user can quickly swipe between pages like a book, without having to go back to the menu to select a new page.

12 Technical Reviews

12.1 Christopher Hebert

Oregon State University
College of Engineering

CS Capstone Final Report

June 12, 2018

Music Theory Application

Prepared by

Aidan O'Malley

Signature Date

Abstract

This document outlines several technology-related design decisions specifically about the composition page. Methods for chord entry, chord editing, and composition analysis are discussed. Decisions are made about which technologies to choose.

CONTENTS

LIST OF FIGURES

13 INTRODUCTION

This document evaluates different design and technological decisions for chord entry, composition editing, and analysis. This app will exist in a saturated market, so there are plenty of examples of how other applications handle this set of design choices. Sibelius 7 is a popular application to design musical scores. It allows for chord entry using plain-English text entry, direct MIDI note entry, and it can also calculate chord name and quality from a group of notes. [4] GarageBand is an app for creating music on iOS devices. It allows for chord entry using a Chord Wheel. [3] Chord! is an iPhone and

Fig. 1. Image of GarageBand's Chord Wheel from [3].

iPad app that is designed to help with music composition. Chord! is interesting because it uses analysis to help the user enter chords. Chord! claims that it "uses [music theory] to give more meaningful and accurate results in a way that you can understand too the underlying theory." [2] All of these apps are designed for writing and composing music, and

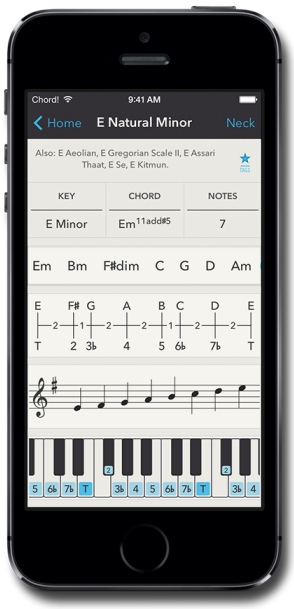


Fig. 2. Image of Chord!'s music theory analysis from [2].

have tools specific to the composition style for editing chords individually. Sibelius 7 and GarageBand allow users to edit chords by editing the notes. Chord! allows users to edit chords individually using a combination of analysis, radio buttons, and check boxes.

14 Composition Page: Chord Entry

14.1 Overview

In order for users to create compositions they need to be able to enter chords. These chords will be based on a root tone and need to have a specific quality. The mechanism for chord entry should be able to handle: choosing a root tone from one of the 12 tones, choosing minor or major chord quality, and choosing whether the 7th is present and its quality. Intervals other than the 7th are outside the scope of this application.

14.2 Criteria

The criteria for choosing which chord entry style to use will include: accuracy of user input, ease of recovering from mistakes in user input, cohesion with the other parts of the app, complexity of implementation.

14.3 Potential Choices

14.3.1 Keyboard Style Chord Entry

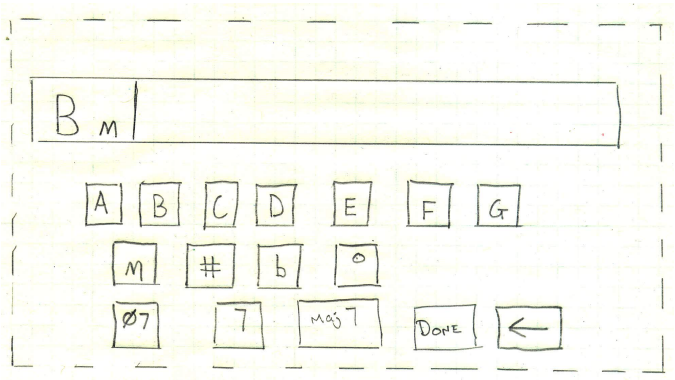


Fig. 3. User in the process of entering “Bm7” using keyboard-style chord entry.

The keyboard style chord entry would have a text area for writing out the chords, as well as a grid layout of buttons for chord entry. The grid of buttons would contain note names, qualities, and intervals. For example, if the user wanted to enter ”B minor 7th”, they would push ”B”, ”m”, ”7” buttons in that order, and the text area would be populated with ”Bm7”.

14.3.2 Circle of Fifths Style Chord Entry

The circle of fifths style chord entry would use a circle of fifths for tone entry, as well as two groups of radio buttons for chord quality and 7th interval quality. The user would touch any of the notes on the circle of fifths to determine the root tone of the chord. The chord quality radio buttons would be either major or minor. The 7th interval quality could be none, 7th, major 7th, or diminished 7th.

14.3.3 Grid of Radio Buttons

The third choice uses: a set of radio buttons to determine tone, a set of radio buttons to determine chord quality, and a set of radio buttons to determine the presence and quality of the 7th.

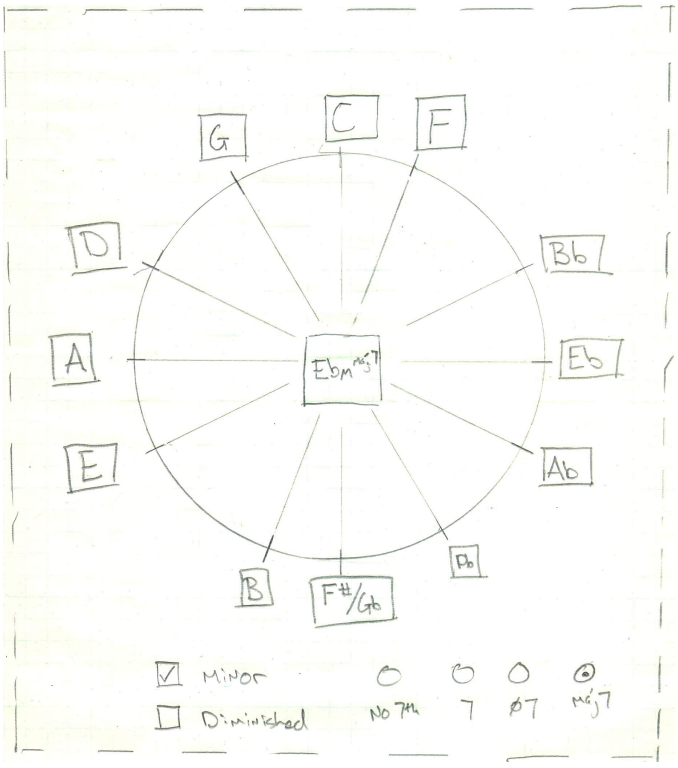


Fig. 4. User has chosen the chord “Ebm Maj7” using the Circle of Fifths-style chord entry.

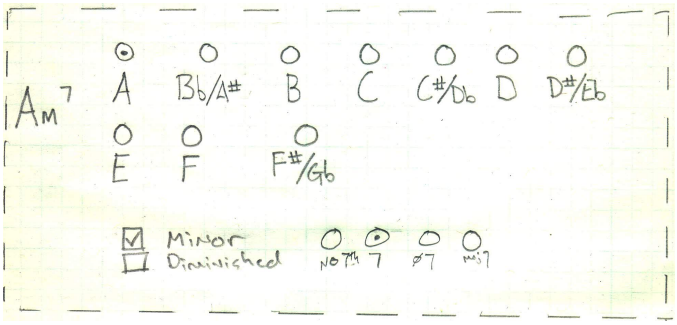


Fig. 5. User has entered the chord “Am7” using the grid of radio buttons-style chord entry.

14.4 Discussion

All three methods allow the user to input chords accurately and with a preview. Keyboard-style entry could allow for invalid chords to be entered, and therefore requires the most work when recovering from user input error. The other two methods don’t allow for invalid chord names to be entered. The Circle of Fifth’s method of chord entry has very high cohesion with the other part of the app: the Circle of Fifth’s page. This would allow users to develop muscle and spatial memory of the Circle of Fifth’s as they practice composing. The keyboard style entry would be the most complex to implement. Keyboard style entry would need: a text entry box, a cursor that can navigate in the text entry box, key entry, deletion, and tests to make sure that the chord entered was valid. The other two entry methods only allow valid chord entries, and are made up of radio buttons and check boxes.

We will be implementing the Circle of Fifths style chord entry, since it solves the problem simply and effectively, and

has the added benefit of drilling muscle memory for users, and has high cohesion with other parts of the app.

15 Composition Page: Composition Editing

15.1 Overview

As users create compositions they may realize that they want to make a change to a chord that is already in the composition. Editing chords means that the user starts with a composition that they don't want, and ends up with a composition that they do want (or at least thought that they wanted). Compositions will be short enough to be visible on a single page. Tools to analyze, and edit will also be on this same page.

15.2 Criteria

The criteria for choosing chord editing will be: ease of use, ease of learning, ease of implementation, and how much visual clutter to the interface the editing style adds.

15.3 Potential Choices

15.3.1 Chords are Implemented as Text Elements with Touch Events

Having chords in a composition as text elements gives them the appearance of being plain text. They can be drawn in sequence. In order to make them editable, they will have a touch event registered to them. When the user touches them, the event will trigger the chord entry to begin changing the chord that the user touched.

15.3.2 Chords are Implemented as Buttons

In this scenario the chords are implemented as buttons. It will be obvious to the user that something will happen if they push the chord buttons. When the user pushes a button, chord entry will begin and the user can change the chord that she or he picked.

15.3.3 Chords are Implemented as Text Elements

In this scenario, the chords are implemented as text elements, but they are not directly editable. If the user wishes to change the composition, he or she can remember the composition they were editing and start a new one with the change implemented. There will be a button which will discard the old composition and create a new one, and will be labeled to indicate this.

15.4 Discussion

The first two are the easiest for editing compositions. These give the user the tools to change only one chord at a time. The second and third options are easiest to learn. It is not obvious that text elements are buttons, but it is obvious that something will happen with the user presses one of the buttons. [5] The third option is easier to learn, since the button to change the composition will be labeled. The third option is the easiest to implement. It only has one button which discards the current composition. The other two implementations have about the same complexity of implementation one button or touch event for each chord. The first two options add several new touchable elements. Since touchable elements should be larger, this would increase the visual clutter. [1]

In conclusion we will be implementing the chords as text elements, without direct edit-ability. This choice is the simplest, and matches the goals of the app the best. The purpose of the composition page is to demonstrate the rules

of the schedule of tonal gravity. Once users have a solid foundation in these rules, they won't have much need for the composition portion of the app.

16 Composition Page: Analysis

16.1 Overview

The purpose of the composition page is to analyze user compositions according to the rules of the Schedule of Tonal Gravity (henceforth referred to as "the rules"). The rules either apply to the first chord, or a transition between two chords. Therefore, analysis requires at most a chord and its subsequent chord to determine a meaningful result. The purpose of this section is to determine trade-offs in when and how much of the compositional analysis is performed.

16.2 Criteria

The criteria for determining the style for analysis include: flexibility of use, ease of understanding, amount of visual clutter.

16.3 Potential Choices

16.3.1 User Picks a Single Chord to Have Analyzed

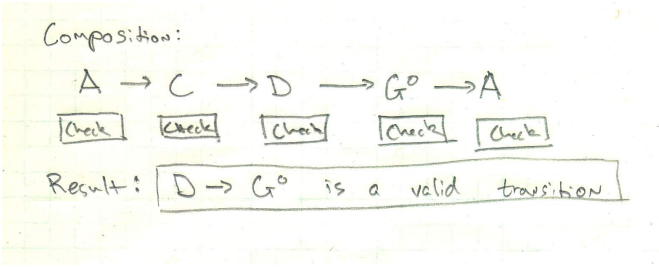


Fig. 6. The user has pressed the "Check" button associated with G diminished in the single chord analysis-style.

In this scenario the user would push a button closely associated with the chord they would like to have analyzed. The chord would be compared to the previous chord (or no chord, in the case of the first chord) to determine whether the chord was a valid choice. There would be a result text-area to show the result of the single analysis.

16.3.2 Entire Composition is Analyzed

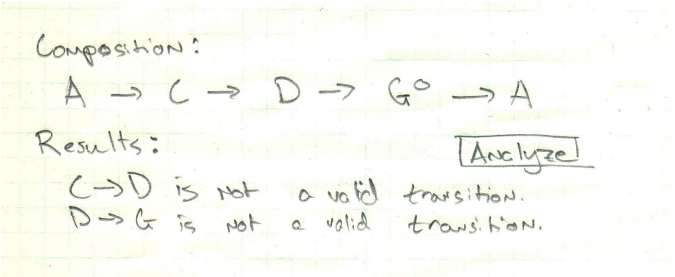


Fig. 7. The results section is populated with all of the broken rules in the composition in the entire composition analysis-style.

In this scenario, the user would push a button that would analyze or re-analyze the entire composition, looping over each chord and pair of chords to determine which followed the rules. There would be a results text-area which would contain a list of all of the transitions that violated the rules.

16.3.3 Composition is Analyzed when Chord is Added or Changed

If a chord is added, that chord would immediately be analyzed. If a chord is changed, that chord, as well as the subsequent chord, would be re-analyzed. The results text-area would be updated to reflect the new analysis.

16.4 Discussion

The first option is the most flexible. It allows the user to not immediately receive feedback, if they wanted to do their own analysis. It allows the user to specify exactly which chord or chord transition they want analyzed. The second option has less flexibility. The user is able to not receive feedback, but they only have the ability to analyze the whole song at once. The third option provides the user with the least flexibility, forcing them to receive feedback whenever a chord is entered.

The third option is the easiest to understand, since it does not require any special input from the user. All analysis is performed automatically. The second option is the also easier to understand, since there is only one button to analyze the entire song, and the user doesn't need to know about how analysis works. The first option could be considered simpler, since the user would immediately know which chord transition posed a problem.

The first option creates the most clutter, since an additional button would need to exist for each chord in the composition. The second option would create a bit more clutter than the third option, but only a single button.

We will be implementing the entire composition being analyzed at the push of a button (choice 2). This maintains enough user flexibility without cluttering the UI. It is also still very easy to understand.

17 Conclusion

The market for apps of this nature may be saturated, but that doesn't mean that we don't need to do our own design work. The trade-offs we have made for our design decisions are very much specific to the app we are designing. We chose the Circle of Fifths method of chord entry to mesh with the main page and thrust of our app. We chose to have compositions be short and simple, to encourage users to start writing down their compositions in their own way. We chose to have editing of compositions be wholesale to give students new opportunities for creativity. We chose to have delayed analysis of composition to give users, who will be using the app to learn music theory, an opportunity to think through the problem themselves. All of these design choices could have been completely different if any of the goals of the app were changed.

References

[1] *UI Design Do's and Don'ts,*
`https://developer.apple.com/design/tips/`

[2] Thomas Grapperon. *Chord! The Definitive Guitar App*
`http://getchord.com/` 2014

[3] *Garage Band for iPad: Add Your Own Custom Chords*
`https://help.apple.com/garageband/ipad/2.3/#/chsab9d1c4c`

[4] *Sibelius 7 Tutorials*
http://hub.sibelius.com/download/documentation/pdfs/sibelius713-tutorials-en.pdf

[5] Nick Babich. *Button UX Design: Best Practices, Types, and States*,
https://uxplanet.org/button-ux-design-best-practices-types-and-states-647cf4ae0fc6?gi=bc6cdbd6bca1
March 15, 2016

17.1 Aidan O'Malley

CS Capstone Final Report

June 12, 2018

Music Theory Application

Lukas Hein _____ _____
Signature *Date*

Prepared by

Group 45
Team TriTone

Aidan O'Malley _____ _____
Signature *Date*

Abstract

This document is a technology review for the Music Theory Application that Team TriTone will be creating for Lukas Hein. This document contains Aidan's review of his three pieces. These pieces include the overall development environment, the base Circle of Fifths implementation, and the Circle of Fifths sidebar implementation. Aidan's role in the group is as a developer and he will help with other general decisions regarding the creation of the application.

18 Overall Development Environment

18.1 Overview

Since our team's task is to create an application, the first hurdle that we must tackle is what environment are we going to define as our base to build the application upon. There are more possibilities than what we are going to discuss in this paper, however, for our problem we have narrowed down our choices of development environment to Xamarin Studio, XCode and Android Studio to create separate but equivalent products, and lastly React Native.

One thing that is important to note is that, no matter what choice of development framework we decide on using, we will need to pay for an Apple developer's license and a Google developer's license to actually release our application. It would be very beneficial if we can limit other costs.

18.2 Criteria

To judge what will determine an acceptable development environment, we need to be able to meet all of the requirements we defined in the requirements document. The main hurdles in that document will be creating a chart to represent the Circle of Fifths, adding interaction capabilities to this chart, as well as creating a page to analyze a user's compositions. The other aspects of the application are mostly simple text based components and should be easily accomplished in any of the development environments we defined.

18.3 Pieces

18.3.1 Xamarin

One environment that our team looked into using is Xamarin. There are many positives to using the Xamarin platform. It is free to use the platform for a small team like ours, this is great because we want to limit the cost of production. Three major benefits of Xamarin are the native user interactions, native api access, and native performance [1]. By native it is meant that when using Xamarin we can get the same user interactions, api access and performance as what we'd get using the development process specific to the device, so Swift for iOS and Java for Android. There is also the positive of using an object oriented programming language. Most of the computer science classes we have taken at Oregon State have focused on object oriented programming so this is a logical way for us to go about programming an application. For the most part, the codebase can be shared between the two platforms we're developing on as well [1]. This is a huge positive because it saves us time and effort with only a three person development team. Lastly there is the benefit of the Xamarin component store and NuGet packages. There are hundreds of implementations from these two sources of all kinds of things from charts to entire gaming libraries like CocosSharp that we can use to meet the criteria of our application [2].

It is important to note that there are a few negatives to the Xamarin platform as well. A big negative is the fact that we would need to code in C#. Although a positive of C# is the object oriented nature that was mentioned earlier, there is also the fact that only one of our team members has worked with this programming language. This may prove a problem if we are constantly struggling with learning a programming language while trying to create an app. Another negative to the Xamarin platform is the difficult manner of creating the user interface. Although it is possible to create good-looking user interfaces in Xamarin, the learning curve for the iOS designer is steep and there is also the trouble of having both an Android UI and an iOS UI [3]. The iOS designer is difficult because each component placed on a page is sized and positioned relative to every other component [3]. This takes a lot of forethought and effort and I have

personally struggled greatly using this user interface designer in previous projects. Lastly, from the research conducted, it was difficult to find a way of creating a chart in Xamarin that would allow us to have all of the functionality that we've deemed necessary to create our Circle of Fifths. To meet all of our requirements we'd have to draw a circle like in the article on Xamarin's website and add all of the other necessary components on top of this [4]. Surely this will prove difficult in implementation, not to mention the circle does not look smooth at all from the example on the Xamarin article.

18.3.2 XCode and Android Studio

Another option we have to develop this application is to split our development completely and focus on Android and iOS implementations separately. Obviously there are tradeoffs that come along with this separation. We will start with the positives that come with using Swift and XCode for iOS development and Java and Android Studio for Android development. Since we are developing in the native environments for both platforms, anything that can possibly done in an Android app can be accomplished and the same can be said for the iOS application [5] [6]. Because of this, all of our requirements from the requirements document can be met using XCode and Android Studio.

There is an unavoidable issue that comes with this selection of development environment, however, which is the fact that all of our work will be doubled. This is because our code is in two entirely different programming languages so the codebases must be separated. Although a lot of the logic might be the same, we can't just copy and paste the same code from one platform to the next. Another unavoidable issue we have with this selection is that none of us have used the Swift programming language before. This was a problem with Xamarin as well since we have limited experience in C#. In the case of XCode and Android Studio, not knowing Swift is even more of a problem because we already have to have two separate codebases, learning a language on top of the extra work we already have might become an insurmountable feat.

18.3.3 React Native

The last option we have determined as a possibility for our application development framework is React Native. There are many benefits to the React Native framework which follow. One benefit of the framework is that it is entirely JavaScript based [7]. This is huge because every team member is very experienced with JavaScript which avoids the trouble of learning a new programming language to begin development. There is also the support of the node package manager or npm. This is a package manager where thousands of libraries are available for use within any project using npm and JavaScript, it is the world's largest software registry [8]. We can use packages from npm like ART and d3 to create our chart and, looking at React Native's documentation, there is clearly enough components in the API to meet the other criteria of our application [9]. The development process is easier with React Native than our other two options in that there is hot reloading features and the ability to develop and test code on a device over wifi [7]. Hot reloading means that whenever changes are saved in the source code, the application that is being tested on a device immediately reloads to reflect those development changes on the device. Another big benefit with React Native is the fact that there is one shared codebase for both platforms, this will obviously save our team a great deal of time. Lastly, since React Native is based on the web development process and is coded in JavaScript, the codebase can easily be transformed into regular React code and can then be deployed as a web application. This would allow our app to reach a greater audience with ease.

The biggest negative that comes with React Native is that the framework is so new that there are not all of the built in features that the native environments or that Xamarin has. Another negative is that there will be a lot of setup to ensure that we can complete all that is in our requirements document. What is meant by this is we will need to make use of lots of third party packages and there will be lots of code that we won't entirely understand to be able to compile a native application from JavaScript code.

18.4 Discussion

It is clear to our group that two of the three pieces above have issues that are very serious and the issues mostly stem from the language being used in the frameworks chosen. Learning a new language to complete our applciation would be an achievable feat if we were sure that the framework would allow us to make the best application possible. This is just not the case, however. It isn't clear where Xamarin would have a leg up over the React Native development environment even if we don't consider the troubles of learning C#. In a similar vein, using Swift/Xcode and Java/Android Studio doesn't offer enough benefits over React Native. Not only would we have to learn Swift to complete our application, but we would also have to work on two separate codebases and likely come out with a project that is half as complete as a project using only one codebase.

18.5 Conclusion

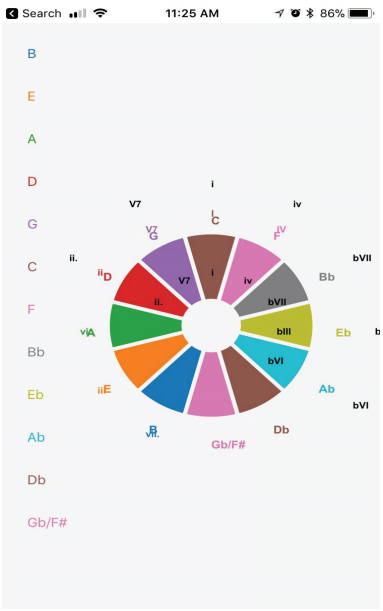
We decided on using React Native as our development environment for many reasons. One reason being the wide range of free third party libraries. Another being the fact that it is one codebase for both platforms. Lastly, there is the least boundary to start with React Native since every team member is comfortable with coding in JavaScript. It is clear that we can meet all of our requirements with this choice of development environment and it will allow us to achieve a finished product with the least amount of difficulty.

19 Base Circle of Fifths Implementation

19.1 Overview

The main part of our application is going to be the Circle of Fifths page. This is what demonstrates most of our client's topics that he wants to teach. It is a circle with the 12 tones of western music around it. Going clockwise around the circle are the intervals of four and going counter clockwise are the intervals of five. There are many ways we could implement this circle but we have narrowed our options down to creating the circle with something similar to a browser SVG (vector based graphic), loading specific images into the app and cycling through these when a user interacts with the app, or lastly using ascii art to make up the circle.

Rotation is a big part of how our circle will work and thankfully rotating components is very easy in React Native with the ability to put a transform property over any group of components [10].



In this picture you can see an example of what we want the Circle of Fifths to potentially look like.

19.2 Criteria

The criteria we defined to determine if the Circle of Fifths is acceptable are listed. It needs to be easy to make out all of the notes on the circle, it should also be easy to understand the flow from relative key to main key to parallel key on the circle. The circle should be visually pleasing to the user meaning it matches the rest of the theme and doesn't stand out negatively.

19.3 Pieces

19.3.1 SVG

Using a vector based graphic that is drawn within the app is one idea we have for implementing the Circle of Fifths. There are many positives in using a graphic that is actually created within the application's source code. The main reason being we have more customizability over the animations and interactions with the circle. Since it is vector based, there will be no issues with resizing the graphic if necessary.

There are also a lot of different options for creating vector based graphics in React Native. This is a positive and a negative because the reason there are so many different libraries to help with creating a vector based graphic in React Native is because SVGs aren't supported by the React Native framework itself. Many people have implemented their own ways of creating these graphics and have put them on the npm registry for others to use, one example is react-native-svg [11]. There are many resources and articles out there explaining ways of making svg charts with libraries like ART and d3 such as the article written by David Vacca [12]. Reading this article brings out some more negatives of the SVG drawing because of the tedious setup required to be able to draw SVGs with the ART library in iOS. However, the ease of adding animations is evident in the article and is a major positive because our application could be improved with good animations. We have a simple application so if we can make all interactions and visuals look slick and clean we can make a really nice app.

19.3.2 Image

Using images such as .png or .jpeg formatting is another option we have to create the Circle of Fifths segment of our application. By using an application like Adobe Illustrator or even something as simple as Microsoft Paint we are able to create a circle that looks however we want and is easy to implement in the application. React Native has a component for images so we can just drop the images in a folder in the source code and load them into the app when an interaction deems an image necessary [13]. All we need to do then is have a transform property on the image component to rotate it and we have a working application.

Some negatives of the image route of implementation is the fact that for each note positioned at the top of the circle, there will need to be six variations of content. There needs to be the major key, the minor key, as well as the parallel and relative keys for both of the main keys. Every combination of those options comes out to six images for each of the 12 notes. This comes out to 72 images to complete the circle which could potentially take up a lot of space on a phone and is a very clunky implementation for this reason. To add to this negative, there will also be no way to smoothly animate between images other than a fade in or fade out. This is because unlike with a vector based graphic where each component is it's own piece of code, the whole image is a piece of code and can't be transformed other than with a rotation of the whole image.

19.3.3 Ascii

The last possible implementation we have thought up for the Circle of Fifths is using ascii art. What this means is using plain text to create a circle and the user can then interact with this text. The implementation would be very easy since we would be able to use React Native's built in Text component [14]. We would then need to use basic circle geometry to place the text. We can then group all of the text under one group component and perform rotations on that. We can also animate text changes because each text component is separated from the group.

The negative to using ascii art or plain text is that it will look very unappealing to the user. Since our application isn't very complex, sacrificing appearance for ease of implementation may not be the best idea.

19.4 Discussion

Overall, it is clear that we have two implementations that are easy to code but have other important limitations. We also have an implementation that is harder to set up and work with, but this implementation gives us a wide range of customization possibilities. With an SVG image we can do everything we could possibly want with our circle in terms of appearance and animation, however, since it is not supported by React Native itself we will have difficulties setting up libraries and working with components. With a normal image we have everything necessary to make a beautiful circle and easily code the interactions, however, we can't interact with specific pieces within the circle and having 72 images in the app takes up an unnecessary amount of space. Lastly with plain text ascii art we can easily create a circle and interact with specific components within the circle as well as the whole circle, however, the appearance of such a circle looks unprofessional and this can't be ignored.

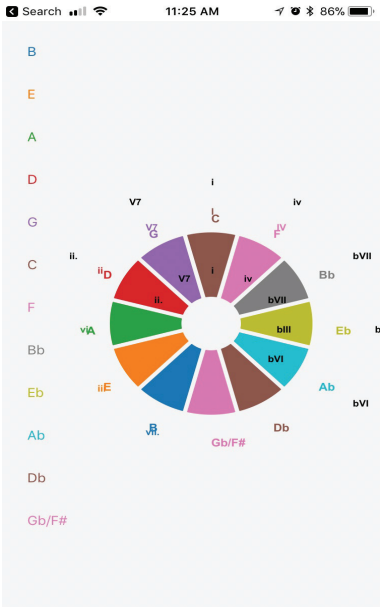
19.5 Conclusion

In conclusion, we have decided to go the SVG vector based graphic route to implement our Circle of Fifths. We recognize that our application as a whole is not that complex so we can take the time to code and interact with a difficult SVG Circle of Fifths if it will allows us to have the best looking main section of our application.

20 Sidebar Circle of Fifths Implementation

20.1 Overview

Since the main part of our application is the Circle of Fifths page, it is necessary to talk about our implementation of the sidebar for the Circle of Fifths. The sidebar's main purpose is to show the user the schedule of tonal gravity for the key that is currently selected. To show the tonal gravity we have decided that the sidebar will be a vertical list of notes where the tonal gravity flows naturally downward. The second purpose of the sidebar is to alter the contents of the circle, for example, the sidebar should allow the user to change the current key in some manner. This is where we have multiple ways of implementing the sidebar's functionality.



In this picture you can see an example of what we want the Circle of Fifths' sidebar to potentially look like.

20.2 Criteria

The criteria we defined for a working sidebar is the following. The user should be able to set the main key with the sidebar. The user should be able to view the tonal gravity of the current key. The user should be comfortable with the gestures used to alter the circle with the sidebar.

20.3 Pieces

20.3.1 Touchable Notes

Our first possible implementation is a vertical list of notes where you can select any note and change the contents of the circle. The change would just be taking whatever note was selected and then moving it to the top of the circle with this note's major key reflected by the chord qualities around the circle. This is an easy implementation using React Native's Text component. The Text component has an "onPress" property which allows us to call a function when text is pressed [14]. This is the exact functionality we want because the function we call when a note is pressed can then handle all of the circle transformations.

The negative to having touchable notes is that a single letter of text may be hard for some people to click on and there could be many issues with selecting the wrong note. Another negative to the touchable notes is that there is not much we can do to transition well from one note's key to another's other than rotating the circle after a click or just resetting the circle immediately after a click. Rotating might annoy users that want to immediately see the changes reflected, however, immediately changing the circle's contents could also be too abrupt for some users.

20.3.2 Draggable Bar

Another implementation is a vertical list of notes where the user can slide their finger from top to bottom on the sidebar and the circle will alter correspondingly. This is a little more difficult of an implementation but not by much. We can group the notes which will be made with the React Native Text component with a View component [15]. On this view component we can then have a React Native PanResponder to react to a user's gestures [16]. In the PanResponder's "onPanResponderMove" function we then include the necessary functionality of altering the circle and we have a complete implementation.

The negative to the draggable bar implementation is that the spacing of the text will determine how easy the dragging gesture is. If the text is too close together, the dragging will rotate the circle too fast. However, for most screen sizes, the optimal distance between each note won't be achievable for a really smooth circle and sidebar combination. Another negative to the draggable bar is that we will need to have a way to lock a note to the top of the circle when the user stops dragging. This is not a huge negative but it is more implementation that is necessary for the dragging to function correctly.

20.3.3 Combination of Click and Drag

The last possible implementation we thought of for the sidebar on our main Circle of Fifths page is to combine both the draggable bar and the touchable note implementations. Obviously this implementation doubles the amount of work necessary for our team, but there are many positives that come along with this. The main positive being that users will be able to have options for altering the circle's contents and can end up focusing on one specific implementation based on personal preference. If one user's fingers are too big to click the notes, they have the option to drag the bar. If one user likes the dragging animation more than clicking then they can drag. If a user likes to see their changes reflected on the circle immediately they can click a note. User's need to be able to be comfortable with our app and giving options on how to use the app would be a big help in achieving comfort.

There is the possibility of the two gestures clashing which could cause problems in the app. If the implementation is clunky and the gestures are constantly cancelling eachother out, then personal preference doesn't matter because the intended functionality is not truly there.

20.4 Discussion

It is clear that there are small issues with all of the implementations listed. With both the touchable notes and the draggable bar, how the notes are presented play a big part in how effective the implementation is going to be. The implementation that combines both touching and draggings allows users to decide which implementation is more effective but there is the issue of the gestures cancelling eachother out and causing problems.

20.5 Conclusion

In conclusion, we decided to go with the combination of clicking and dragging to give the user the most interaction with the Circle of Fifths. We can use the PanResponder to implement both implementations instead of using the Text's ”onPress” property and this should avoid any overlap between the gestures. Since the main problem with the combination implementation can be eliminated we decided we should give the user options with how they want to use our application.

REFERENCES

[1] "Xamarin website," https://www.xamarin.com/platform, accessed: 2017-11-19.

[2] "Cocossharp game development information," https://developer.xamarin.com/guides/cross-platform/game_development/cocossharp/, accessed: 2017-11-19.

[3] "ios user interface," https://developer.xamarin.com/guides/ios/user_interface/, accessed: 2017-11-19.

[4] "Drawing a simple circle," https://developer.xamarin.com/guides/xamarin-forms/advanced/skiasharp/basics/circle/, accessed: 2017-11-19.

[5] "Swift," https://developer.apple.com/swift/, accessed: 2017-11-19.

[6] "Android studio," https://developer.android.com/studio/index.html, accessed: 2017-11-19.

[7] "React native homepage," https://facebook.github.io/react-native/, accessed: 2017-11-19.

[8] "npm homepage," https://www.npmjs.com/, accessed: 2017-11-19.

[9] "Gettin started," https://facebook.github.io/react-native/docs/getting-started.html, accessed: 2017-11-19.

[10] "Transforms," https://facebook.github.io/react-native/docs/transforms.html, accessed: 2017-11-19.

[11] "React native svg github," https://github.com/react-native-community/react-native-svg, accessed: 2017-11-19.

[12] "Animated charts in react native using d3 and art," https://medium.com/the-react-native-log/animated-charts-in-react-native-using-d3-and-art-21cd9ccf6c58, accessed: 2017-11-19.

[13] "Images," https://facebook.github.io/react-native/docs/image.html, accessed: 2017-11-19.

[14] "Text," https://facebook.github.io/react-native/docs/text.html, accessed: 2017-11-19.

[15] "View," https://facebook.github.io/react-native/docs/view.html, accessed: 2017-11-19.

[16] "Panresponder," https://facebook.github.io/react-native/docs/panresponder.html, accessed: 2017-11-19.

21 Weekly Blog Posts: Christopher Hebert

21.1 Fall

21.1.1 Week 1

21.1.1.1 Plans:

- Setup OneNote Outline
- Write Biography
- Update Resume
- Choose Project preferences

21.1.1.2 Problems:

- Wrong OneNote Email

21.1.1.3 Progress:

- Setup OneNote Outline
- Wrote Biography
- Updated Resume

21.1.1.4 Summary: This week I updated and printed of my resume. I needed to remove some projects from my resume that were no longer in-progress and not finished. I updated the language of my resume to reflect my actual skills, e.g. not claiming proficiency in any languages. I also wrote my biography, envisioning what kinds of work and projects I would want to spend the rest of my life doing. Finally, I chose the projects I was interested in: my choices and reasons are listed below.

21.1.2 Week 2

21.1.2.1 Plans:

- Meet with project group
- Contact project client
- Have a meeting with the project client
- Determine project details
- Discuss potential solutions
- Discuss metrics
- Determine IP rights
- Work on problem statement
- Format latex document
- Describe problem
- Propose solution
- Determine metrics
- RD: 10/9/17
- FD (with client verification): 10/20/17

21.1.2.2 Problems:

- App idea is very small right now, and undeveloped

21.1.2.3 Progress:

- Was assigned a client:
- Lukas Hein: lukasalihein@gmail.com
- Met with group:
- Aidan sent contact email to client, CC-ing Kaz and me
- Put agenda and discussion questions in the project notebook
- Meeting in person with group at The Beanery @ 4p 10/5/17
- Discussed music theory, IP, future meetings, and a little bit of the app

21.1.2.4 Summary: This week I met with the client and the rest of the team to discuss the app. We met on Thursday and talked for about 50 minutes. He introduced us to a small amount of music theory, and told us that that was what he wanted the core of the app to be. I tried to press him a bit for more details, but he seemed unsure of what we would be able to accomplish so he left it vague. I put together some sketches of what I thought could be some of the app pages, based on what we talked about, and a couple of ideas I had of my own. Next week I hope to show Lukas the sketches and see what he thinks of them, and perhaps put together some digital prototypes to convey the idea better. This week we also set up communications. I set up a Slack channel for us to chat, and Kaz set up a Github project page along with a Waffle.io kanban board. In addition we also discussed IP rights. Lukas wants 40% of the share, since he is providing the decades of music theory knowledge. We informally agreed to a 20-20-20-40 split between the 4 of us, although no documents were signed and no concrete assent was given. Since my main focus is the class project and graduating, I made it clear that I wasn't interested in handling any of the business side of things. If I get any money from this, great. If I don't, great I don't really care. I just want to make a good app.

21.1.3 Week 3

21.1.3.1 Plans:

- Meet TA Tuesday
- Meet Lukas Thursday
- Make prototype of sound waves app page

21.1.3.2 Progress:

- Met TA Tuesday
- made a prototype of the sound waves idea I had
- Met with Lukas
- Made a simple prototype of the composition page
- Drew paper prototypes of parallel/relative minors

21.1.3.3 Summary: This week was a productive week. I made some small, simple prototypes using Javascript with the hopes of assuring the client that we were serious, inspiring teammates to come up with their own ideas, and play around with the theory I learned from our meeting last week. Some interesting things I learned prototyping

the sound waves idea: Just intervals are based on simple ratios of sound waves, e.g. 3:2 is a perfect 5th Just intervals do not perfectly complete a circle of 5ths Instead we use equal temperament intervals, stretching the intervals to make everything fit nicely. Based on my composition prototype, Lukas mentioned that chord entry would be a critical component. He showed us another music theory app that presents a chord keyboard to edit chords.

21.1.4 Week 4

21.1.4.1 Problems:

- IP distribution and IP for stretch goals
- Problem statement

21.1.4.2 Progress:

- Met with client
- Met with TA
- Sketch of parallel/relative minor keys

21.1.4.3 Summary: This week the client renegotiated from 40-20-20-20 split to a 25-25-25-25 split. I think Lukas was starting to see how much work and design needed to be done in order to convert his idea into an app, and felt that an even split would be more equitable. We discussed retaining rights for stretch goals, and he was on board for this. I made a couple of more sketches based on what Lukas taught us about transitioning to parallel and relative keys. The next step is coming up with storyboards about how the app can be built. I also came up with an idea about how chord entry could be done using the circle. This week I asked Lukas about how the app was intended to be used. Specifically, I wanted to know if the controls needed to be designed for musicians who were playing their instruments while using it.

21.1.5 Week 5

21.1.5.1 Plans:

- Meet with TA Tuesday
- Meet with Lukas Thursday
- Start working out app flow and specific page layouts and designs
- Work out high level connections (which pages lead to what)
- Make a rough draft of the Requirements Document

21.1.5.2 Problems:

- What is the app flow?
- What is the circle of 5ths page layout?
- How many substitutions shown at once
- How to show parallel/relative keys
- How to engage/disengage parallel keys
- How to swap between substitutions
- What is background info layout?

21.1.5.3 Progress:

- Came up with a prototype for circle of 5ths
- Met with TA. Discussed IP
- Met with Kirsten. Discussed IP
- Broke up requirements subsections
- Wrote requirements document rough draft

21.1.5.4 Summary: This week several stories threaded together. From last week, Lukas wanted to start working towards an IP contract. Unfortunately, after discussing it with Kirsten the process seems potentially tricky, so Lukas will need to take the baton on this one. This week we have been working on nailing down the UI interactions and user stories. There is a lot of information that we need to present in a very small amount of screen space, so the problem may be trickier than everyone imagined. Aidan volunteered to make an interactive prototype to see how much we could reasonably fit on the iPhone at once.

21.1.6 Week 6

21.1.6.1 Plans:

- Tue: Meet with TA
- Finish requirements doc
- Send rough draft to Lukas
- Thur: Meet with Lukas

21.1.6.2 Problems:

- Need visualizations
- how do they fit in to the latex?
- What quality? (hand drawn?)
- Need a Gantt chart

21.1.6.3 Progress:

- Met with TA
- Met with Kirsten to discuss rough draft

21.1.6.4 Summary: This week was a lighter week. We had the requirements nailed down last week, we also started design work from the beginning of the quarter. I met with Kirsten to discuss what kinds of visualizations might be needed for the requirements document, only to find out it was a misunderstanding of her note to our rough draft. We discussed what the requirements document needed to be: only content, not any particulars about UI design. Aidan made changes accordingly, and we submitted the final draft to Lukas for review and signature. He signed and sent the email with a flourish.

21.1.7 Week 7

21.1.7.1 Plans:

- Meet with client to discuss prototype
- Meet with TA
- Split up tech review problems

21.1.7.2 Progress:

- Aidan's prototype
- Aidan split up the tech review pieces

21.1.7.3 Summary: This week we met with the client to talk about Aidan's prototype. We played around with different font sizes and talked about color schemes and visuals. We discussed time travel and trips to Morocco, jazz drummers who use paint brushes instead of sticks, and where to find definitions for our requirements document.

21.1.8 Week 8

21.1.8.1 Plans:

- Rough draft of tech review
- Get set up with react native
- Work on composition prototype using react native
- Review "escape the lab"

21.1.8.2 Summary: This week I finished the rough draft of the tech review and talked to Kirsten about how to implement it. I didn't have enough time to complete the composition prototype, so we canceled our meeting with the client.

21.1.9 Week 9

21.1.9.1 Plans:

- Review escape the lab
- Final draft of tech review
- thanksgiving

21.1.9.2 Summary: This week's class/client/TA meetings were canceled due to Thanksgiving holiday. I submitted my final draft of the tech review. Aidan, Kaz, and I did the Escape the Lab at the MU and I sent a review to Kirsten for extra credit on the tech review.

21.1.10 Week 10

21.1.10.1 Plans:

- Complete preliminary design doc
- Complete progress report
- Final meeting with TA

21.1.10.2 Summary: This week was another light week, since we were all pretty well prepared for our design document. A lot of what we have been doing up to this point in the quarter has been design related, and we already have a couple of prototypes so we are familiar with implementation.

21.2 Winter

21.2.0.1 Plans:

- Attend class
- Get Aidan's prototype working on my computer/iPhone

21.2.0.2 Problems:

- Working with Node & NPM version problems

21.2.0.3 Progress:

- Hello World app working
- Got Aidan's prototype working

21.2.0.4 Summary: This week we were getting back into the swing of things. I got Aidan's prototype working on my iPhone and computer by setting up Node JS, NPM, React Native, and Expo. The prototype looks great.

21.2.1 Week 2

21.2.1.1 Plans:

- Divide up work
- Reschedule/Re-evaluate what we will do now that Kaz is gone
- Set up meeting schedule with TA
- Set up meeting schedule with client (Lukas)

21.2.1.2 Problems:

- Kaz no longer a part of the group

21.2.1.3 Progress:

- Meet TA W 1:30-2
- Meet Lukas 2:15 -

21.2.1.4 Summary: I met with Kirsten on Tuesday to talk about how Kaz's departure would affect the team and our workload. We met with Behn and Lukas on Wednesday, and set up regular meetings. We discussed our winter term goals with Lukas, as well as how he could contribute his music theory knowledge to our reference pages.

21.2.2 Week 3

21.2.2.1 Plans:

- Add start to reference pages

21.2.2.2 Problems:

- Learning new version of JS is a headache
- Learning libraries of React is overwhelming

21.2.2.3 Progress:

- Created a start to reference pages

21.2.2.4 Summary: This week I got the bare minimum in for reference pages. It's basically a "Hello World" page. With minimal styling. The team met with Behn and the client on Wednesday to discuss our phenomenal progress.

21.2.3 Week 4

21.2.3.1 Plans:

- Add markdown to reference pages
- Work on composition page

21.2.3.2 Progress:

- Added markdown
- Worked on composition page

21.2.3.3 Summary: This week I fleshed out the reference pages. I added markdown support so that Lukas could write the reference pages himself without learning to code. I also built a skeleton for the composition pages. It all operates, but it doesn't quite do anything useful.

21.2.4 Week 5

21.2.4.1 Plans:

- Work on analysis
- diatonic subs
- borrowed (parallel)
- secondary dominants
- altered dominants
- diminished triads
- layers for analysis:
- allow different substitutions
- remove 7ths
- replace augmented with dominant

21.2.4.2 Problems:

- None on my end, yet

21.2.4.3 Summary: This week I worked on analysis. I thought about diatonic subs, borrowed, parallel, and altered dominants. I met with the client and Aidan and our TA.

21.2.5 Week 6

21.2.5.1 Plans:

- Work on Analysis page (tonal gravity)

21.2.5.2 Progress:

- Organized the way things would look
- Use text as a standin

21.2.5.3 Summary: This week I worked with Aidan and Lukas and Behn to further the progress of our app.

21.2.6 Week 7

21.2.6.1 Plans:

- Work on analysis (tonal gravity)
- Work on choosing root tone/quality

21.2.6.2 Progress:

- Organized analysis page
- Tonal gravity theory
- More state in component

21.2.6.3 Summary: This week we did not meet with Lukas. I talked to Behn about what he was doing for his final project for his graphics class. We talked a little bit about the lack of progress we've made. I'll make more progress next week.

21.2.7 Week 8

21.2.7.1 Plans:

- Meet Lukas
- Meet Behn
- Think about how to break up analysis page

21.2.7.2 Progress:

- Met with lukas
- Met with Behn

21.2.7.3 Summary: This week in our meeting with Lukas we talked about how to organize some of the introductory material for the reference pages. He wanted to keep his words simple enough for everyone to understand, but his vocabulary is quite large, and we needed to tone down some of his artistic speech. We discussed the possibility of augmenting some of the reference pages with toys and demos of what Lukas is describing to better cement the concepts in our future composers' brains.

21.2.8 Week 9

21.2.8.1 Plans:

- Get analysis page working
- See what Lukas thinks of how analysis is displayed

21.2.8.2 Problems:

- Debugging in react native is frustrating

21.2.8.3 Progress:

- Got analysis page working

21.2.8.4 Summary: This week I got the analysis page working for the app. This includes setting a Key for the composition, creating chords, and having the computer analyze them according to the rules of Tonal Gravity. I also included a bit that shows roman numeral analysis if the user is using chords in the diatonic key.

21.2.9 Week 10

21.2.9.1 Plans:

- Make final presentation
- Work on written report
- Incorporate Lukas' reference pages

21.2.9.2 Problems:

- Markdown does not work
- Need a scroll-view for reference pages

21.2.9.3 Progress:

- Replace markdown with simple text
- Made final presentation

21.2.9.4 Summary: This final week Aidan and I wrapped up what we needed to for the beta and our final presentation. I put Lukas' reference pages together and in Markdown format. Unfortunately, the Markdown engine was spitting out bad formatting, so for demo purposes we left them unformatted as plain text.

21.3 Spring

21.3.1 Week 1

21.3.1.1 Plans:

- Create a developer account
- Create a build for iOS
- Release to iOS

21.3.1.2 Problems:

- Paying to create a developer account
- Review for app store

21.3.1.3 Progress:

- Set time for client meeting
- requested time for TA meeting.

21.3.1.4 Summary: I had a busy Spring break, as I had traveled to Beijing, so I didn't have any time to work on the app. I spent this week getting back into the swing of things, and readjusting my internal body clock.

21.3.2 Week 2

21.3.2.1 Plans:

- Release to app stores

21.3.2.2 Progress:

- Spent more time testing and tweaking.

21.3.2.3 Summary: This week we started talking about adding chord substitutions back into the composition page. Lukas described in good detail how these should work out. We worked on coming up with compromises for detecting complicated substitutions.

21.3.3 Week 3

21.3.3.1 Plans:

- Work on substitutions

21.3.3.2 Problems:

- Which substitutions to support?

21.3.3.3 Summary: This week was focused more on Circle of Fifths. I spent a lot of time procrastinating.

21.3.4 Week 4

21.3.4.1 Plans:

- Implement chord substitutions
- Match color scheme of Circle of Fifths page for Composition page
- Implement reference pages.

21.3.4.2 Problems:

- The Markdown version of reference pages looks terrible.

21.3.4.3 Progress:

- Lukas made reference pages

21.3.4.4 Summary: The good news was that Lukas finally made the reference pages. The big problem this week was the Markdown not looking so great. This meant that we had to format things ″by hand″ using the React Native markup syntax. I again put off implementing chord substitutions.

21.3.5 Week 5

21.3.5.1 Plans:

- Implement chord substitutions
- Implement reference pages

21.3.5.2 Problems:

- Cannot implement diatonic substitutes

21.3.5.3 Progress:

- Implemented chord substitutions.

21.3.5.4 Summary: This week I finally got around to implementing the algorithms for analyzing chord substitutions. I ran into the issue of not knowing when a diatonic substitute was taking place. They are extremely difficult, if not impossible, to detect because they look exactly like unsubstituted chords. I will talk to Lukas about this next week. Additionally I colored the substitutes and diatonic chords to match the colors in the Circle of Fifths.

21.3.6 Week 6

21.3.6.1 Plans:

- Discuss diatonic substitutes
- Discuss what's next
- Upload to app store.
- Create a developer account

21.3.6.2 Problems:

- Who is going to own the developer account?
- I don't own a Mac

21.3.6.3 Progress:

- Discussed the diatonic substitutes
- researched uploading to app store

21.3.6.4 Summary: I showed Lukas the new composition page and he seemed happy with it. We talked about how it was probably impossible to do diatonic substitutes, and we would include some info about that in one of the reference pages.

21.3.7 Week 7

21.3.7.1 Plans:

- Create an iOS build
- Create a Developer account
- Push the app to the iOS store.

21.3.7.2 Problems:

- iPad did version did not work

21.3.7.3 Progress:

- Created a developer account and pushed the app to the store!

21.3.7.4 Summary: This was an exciting week. We finally got to see our app uploaded to the store! The process took a couple of days, because it was in review. Unfortunately, for whatever reason, the iPad version didn't work even though it worked in the simulator. However, it was good enough for Expo.

21.3.8 Week 8

21.3.8.1 Plans:

- Get ready for Expo
- Make the app free for expo

21.3.8.2 Problems:

- Bugs on Android
- Can only give out promo codes

21.3.8.3 Progress:

- Showed off our hard work to the wonderful people of Corvallis.

21.3.8.4 Summary: This week was Expo. We worked out a way to make the app free for a day on Play and iOS, so that we could share it with people walking by. We refined our pitch as the day went on. We had a keyboard out, but people thought it was part of the app, so we hid it about halfway through.

21.3.9 Week 9

21.3.9.1 Plans:

- Fix color scheme mismatch
- create a version that works on iPad.
- Work on final presentation and report.

21.3.9.2 Summary: We talked to Lukas about what to do next. I really want to see more interactivity in the reference pages, as well as progressive exercises. I also want some audio elements, even if it's just sound effects for button clicks.

21.3.10 Week 10

21.3.10.1 Plans:

- Release application to iPad
- Fix color scheme mismatch
- Finish final report

21.3.10.2 Progress:

- Released application and first update to iOS

21.3.10.3 Summary: This week we made our first update to the iOS store. This was again a learning process, but it was very similar to uploading for the first time: including signing contracts and logging in via Xcode.

21.4 Weekly Blog Posts: Aidan O'Malley

21.5 Fall

21.5.1 Week 1

21.5.1.1 Plans:

- Start thinking about how I'd want to develop the music application project.

21.5.1.2 Problems:

- No problems yet

21.5.1.3 Progress:

- Messaged Lukas Hein about working on his project.

21.5.1.4 Summary: Saved my preferences after thinking about what projects I was interested and which ones I wasn't. Also contacted Lukas Hein who submitted the project related to the interactive music theory app. I emailed him on Thursday, however, and as of writing this I haven't heard a response.

21.5.2 Week 2

21.5.2.1 Plans:

- Talk to my group.

21.5.2.2 Problems:

- None so far.

21.5.2.3 Progress:

- Was assigned a project!

21.5.2.4 Summary: Was assigned my project (Interactive Music Theory App) and my group and I met up with our client to discuss our project in further detail. Also worked on the problem statement assignment this week and thought deeply on what the application is meant to accomplish.

21.5.3 Week 3

21.5.3.1 Plans:

- Schedule regular meetings with Lukas.

21.5.3.2 Problems:

- None so far.

21.5.3.3 Progress:

- Have a more complete idea of what the app might be.

21.5.3.4 Summary: Met with TA Behnam Saeedi on Tuesday, met with client on Thursday to talk about the scope of the app. Fleshed out some more ideas and have a more complete outlook for what the app will be.

21.5.4 Week 4

21.5.4.1 Plans:

- Think of presentation of application/mock ups
- feature list, metrics, etc.
- okay to have some flexible requirements, etc.

21.5.4.2 Problems:

- Not yet

21.5.4.3 Progress:

- Nailed down requirements

21.5.4.4 Summary: Finished the problem statement as a group after discussing more of how we want to finished product to turn out. Met with ta and took notes. Talked about ip issue. Met with Lukas and nailed down app requirements and Lukas brought up splitting the ip evenly four ways so that issue solved itself.

21.5.5 Week 5

21.5.5.1 Plans:

- have prototype app for next week

21.5.5.2 Problems:

- None.

21.5.5.3 Progress:

- template app started.
- requirements rough draft.

21.5.5.4 Summary: Worked on creating template app in react native. Met with Lukas and hammered down some user interactions. Finished requirements doc rough draft.

21.5.6 Week 6

21.5.6.1 Plans:

- Continue working on prototype

21.5.6.2 Problems:

- None

21.5.6.3 Progress:

- Visible pie chart

21.5.6.4 Summary: Worked on prototype of circle of fifths. Got environment set up in react native and have a pie chart visible using d3 and ART third party libraries. Also finished requirements document.

21.5.7 Week 7

21.5.7.1 Plans:

- Complete more of circle of fifths prototype.
- Work on design doc more

21.5.7.2 Problems:

- Where to put tritone, diatonic, diminished subs in circle
- Where to put secondary dominant in circle

21.5.7.3 Progress:

- spinning working and rotating labels work on prototype
- Prototype is in GitHub under app-source-code folder

21.5.7.4 Summary: Met with client and went over design goals after showing prototype. Worked on prototype for circle of fifths, now have spinning working and rotating labels. Worked on splitting up Tech Review.

21.5.8 Week 8

21.5.8.1 Plans:

- Finish tech review and have more for Lukas to critique with the prototype

21.5.8.2 Problems:

- Piece 3 of tech review doesn't have great sources or implementation options.

21.5.8.3 Progress:

- Worked on tech review

21.5.8.4 Summary: Worked on tech review and prototype using notes taken from meeting with Lukas during week 7.

21.5.9 Week 9

21.5.9.1 Plans:

- Have more for Lukas to critique with the prototype
- Work on Design Doc
- Work on Progress Report

21.5.9.2 Problems:

- None currently

21.5.9.3 Progress:

- Finished tech review

21.5.9.4 Summary: Worked on tech review and finished it. Started thinking about Design Document and how to begin work on that.

21.5.10 Week 10

21.5.10.1 Plans:

- Have more for Lukas to critique with the prototype
- Work on Progress Report
- Work on application source code

21.5.10.2 Problems:

- None

21.5.10.3 Progress:

- Finished Tech Review

21.5.10.4 Summary: Finished my part of the Design Document and considered more implementation options for the circle of fifths

21.6 Winter

21.6.1 Week 1

21.6.1.1 Plans:

- Figure out how to handle not having Kaz anymore.

21.6.1.2 Problems:

- Kaz is no longer in capstone

21.6.1.3 Progress:

- Added redux to prototype to handle state management.

21.6.1.4 Summary: Set up redux to manage the state of the application.

21.6.2 Week 2

21.6.2.1 Plans:

- Work on menu and adding reference pages before continuing with COF and composition pages

21.6.2.2 Problems:

- None

21.6.2.3 Progress:

- Made more progress with redux state management and static key signature file

21.6.2.4 Summary: This week we met with Lukas for the first time since fall term. We devised a plan to continue with the project without Kaz and discussed the alteration of a few features.

21.6.3 Week 3

21.6.3.1 Plans:

- Work on circle of fifths page and state management

21.6.3.2 Problems:

- Chris doesn't want to use jsx

21.6.3.3 Progress:

- Menu and reference pages started

21.6.3.4 Summary: This week I created the menu for the application and Chris worked on the Reference pages

21.6.4 Week 4

21.6.4.1 Plans:

- We made the plan to use a scroll view with the sidebar so that more notes could be shown in the sidebar.

21.6.4.2 Problems:

- None currently

21.6.4.3 Progress:

- Added reselect and continued restructuring of Circle of Fifths

21.6.4.4 Summary: This week I worked on more of the Circle of Fifths page. I set up reselect to use with the redux state management.

21.6.5 Week 5

21.6.5.1 Plans:

- I will continue to work on the circle of fifths page and start working on getting the application on an Android device.

21.6.5.2 Problems:

- None

21.6.5.3 Progress:

- Being able to show the application on an ios simulator is good progress because now we don't have to rely on the wifi connection to be able to demo the application on a computer.

21.6.5.4 Summary: This past week I worked more on setting up the application with an ios simulator and got that working. I continued my rework of the Circle of Fifths page and confirmed what notes in the sidebar should be clickable as per Lukas.

21.6.6 Week 6

21.6.6.1 Plans:

- Continue my work on the circle of fifths page.

21.6.6.2 Problems:

- None

21.6.6.3 Progress:

- Got the two assignments for this term completed.

21.6.6.4 Summary: This past week Chris and I spent a lot of time thinking about the progress report and poster. There wasn't too much development time available but we got the two assignments in and felt pretty good about them.

21.6.7 Week 7

21.6.7.1 Plans:

- Start implementing new circle of fifths design.

21.6.7.2 Problems:

- We need to catch up with Lukas.

21.6.7.3 Progress:

- Planning progress but no development progress on my side.

21.6.7.4 Summary: This week I finished my plan for altering the circle of fifths from a prototype to a more complete project. We did not meet with Lukas this week because I had to travel to Portland.

21.6.8 Week 8

21.6.8.1 Plans:

- Add the rest of the circle of fifths features, then move to debugging/testing and lastly add reference pages.

21.6.8.2 Problems:

- Styling for different screen sizes will most likely be an issue. This will need to be figured out by getting the android emulator working and testing screen sizes with that as well as the iPhone emulator.

21.6.8.3 Progress:

- Talked to Lukas about a few things to add in the circle of fifths to finish up.

21.6.8.4 Summary: Finished most of the circle of fifths reimplementation. Components have been split up in a more logical manner and the new design makes more sense and utilized more of the screen.

21.6.9 Week 9

21.6.9.1 Plans:

- Start working on rotating the wheel to change the key as well as rotate after changing the key with the sidebar.
- Work at Chris's house next week to code while Lukas is around

21.6.9.2 Problems:

- Still have a lot of little bugs to address

21.6.9.3 Progress:

- Checkboxes done

21.6.9.4 Summary: This week I got working checkboxes into the application. These checkboxes allow a user to show/hide the parallel and relative keys as well as change between major and minor.

21.6.10 Week 10

21.6.10.1 Plans:

- Meet up Sunday to code and work on video

21.6.10.2 Problems:

- Havent started deployment to apple or google store partially because of money and figuring that out between the three of us but also because the app still has plenty of bugs that users shouldnt be exposed to yet.

21.6.10.3 Progress:

- Fully featured app complete

21.6.10.4 Summary: This week I got rotation working when rotating the wheel to select a key and when selecting a key with the sidebar. We had a development session with Lukas in the room on Wednesday which was super productive and we got a lot of questions answered.

21.7 Spring

21.7.1 Week 1

21.7.1.1 Plans:

- Release to app store and google play store

21.7.1.2 Problems:

- Cost to release app

21.7.1.3 Progress:

- Testing

21.7.1.4 Summary: Added some tests and tweaked implementation over break

21.7.2 Week 2

21.7.2.1 Plans:

- Release to app store and google play store

21.7.2.2 Problems:

- None as of now

21.7.2.3 Progress:

- Testing

21.7.2.4 Summary: Started rewriting circle of fifths to keep qualities the same when key changes and only changing notes. Major and minor qualities will be different but this will make the code more extendable.

21.7.3 Week 3

21.7.3.1 Plans:

- Release to app store and google play store

21.7.3.2 Problems:

- None as of now

21.7.3.3 Progress:

- Finished rewriting Circle Of Fifths

21.7.3.4 Summary: Finished rewriting Circle Of Fifths

21.7.4 Week 4

21.7.4.1 Plans:

- Move working source code over
- Create app icon

21.7.4.2 Problems:

- Cost to release app

21.7.4.3 Progress:

- Testing

21.7.4.4 Summary: Tested Circle of Fifths new implementation.

21.7.5 Week 5

21.7.5.1 Plans:

- Release to app store and google play store

21.7.5.2 Problems:

- Bug with being able to scroll to an empty circle
- Bug with minor colors for chord and sidebar not matching up with circle

21.7.5.3 Progress:

- Summary explains most of the progress.

21.7.5.4 Summary: Put working source code into github. The original source code wasn't working on android but this one is. Also created an app icon and finished adding reference pages and fixed the menu. Completed progress reports.

21.7.6 Week 6

21.7.6.1 Plans:

- Start thinking of future implementations.

21.7.6.2 Problems:

- No problems!

21.7.6.3 Progress:

- Fixes bugs that were specified in the problems section last week

21.7.6.4 Summary: Rewrote the chords modal implementation to free up more screen space and make the style look better. Fixed bugs that were standing out.

21.7.7 Week 7

21.7.7.1 Plans:

- Push to ipad

21.7.7.2 Problems:

- No iPad functionality

21.7.7.3 Progress:

- App is on App Store and Google Play!

21.7.7.4 Summary: We fixed the bugs that we specified a few weeks ago and then deployed the application to the google play store and the app store. We then had expo which went really well!

21.7.8 Week 8

21.7.8.1 Plans:

- Fix new bugs that have been found and add inversion page

21.7.8.2 Problems:

- New bugs

21.7.8.3 Progress:

- We're going to continue working on the app.

21.7.8.4 Summary: Met with Lukas and defined what's needed for our next deployment and what our plans are for after capstone is finished. We decided that we're going to continue working on the app until one of us stops being interested.

21.7.9 Week 9

21.7.9.1 Plans:

- Fix new bugs that have been found and add inversion page
- Work on final presentation

21.7.9.2 Problems:

- None

21.7.9.3 Progress:

- None

21.7.9.4 Summary: Met with Lukas and worked some more on defining new features as well as starting coding again.

21.7.10 Week 10

21.7.10.1 Plans:

- Release application to iPad on Monday including bug fixes
- Finish final report

21.7.10.2 Problems:

- Few more bugs to figure out

21.7.10.3 Progress:

- Finished final presentation

21.7.10.4 Summary: Meeting with Lukas gave us some more fixes to make before pushing the application to ipads. We started working on the final report and finished the final presentation.

22 Final Poster

23 Project Documentation

23.1 Structure of Project

The project is structured as a React Native app, and follows best practices as prescribed by the community. The app source code is based in the "fifths" folder of the repository. The top level contains icons and configurations for building and deploying the app. The sub-folder "src" contains the hand-written and React-generated Javascript source code and the "__tests__" folder contains Perfect Fifth's integration and unit tests for the algorithms.

Within the "src" folder, the "components" folder contains the majority of the code written for the app, organized by the pages of the app (Circle of Fifths, Composition, References, and Menu). The "src" folder also has several other

folders: ”actions” has glue code, ”static” has constants that are shared between files, ”util” has a distance calculation, ”reducers/keys.js” has an algorithm for converting between enharmonic equivalents.

The files in ”components” are split up into the pages of the app: Circle of Fifths, Composition, References, and Menu. This breakdown matches the design of the app laid out in the design document. Each of these exports an object of type React Component. For example, ”Composition.js” exports a ”cCompositionEditView” which is what the user interacts with when creating their own compositions. The layout of the page is determined in the ”render” member function of cCompositionEditView, which returns one or more React Elements (analogous to HTML Elements).

Each reference page is self-titled in the ”References” folder. The individual reference pages are combined together into a ”swipeable” list in the ”index.js” file. The pages again are combined together in ”Menu.js” to create a table of contents, using some more glue code.

The component for the Circle of Fifths page is found in the ”index.js” file of the ”Circle of Fifths” folder. We used the ART.js package to be able to draw a pie chart onto a mobile device and the code relating to this can be found throughout the files inside of the COFContainer folder, most specifically in the ”index.js” and the ”Circle/index.js” files. We were then able to make this circle rotate by using a PanResponder from React Native and some geometric logic. The rest of this component involves normal click handlers, React Native Views, and modals which didn’t involve bringing in any new JavaScript code packages. However, this component uses Redux fairly extensively which helps keep the state of the application consistent within other components. For example, the initial current key in the application is ’C major’, however, if the user rotates the circle or clicks the side bar to change the key to ’A major’, we dispatch a Redux action to the Redux reducer which contains logic that Aidan implemented which handles how we should update the application’s state for this specific interaction. Using Redux really helped Aidan logically plan out most of this component’s interactions and structure.

23.2 Installation of Dependencies

To get the app for your iPhone/Android, simply purchase and download ”Perfect Fifth” from the iOS/Play stores published by Christopher Hebert and Aidan O’Malley respectively.

1) Install the node package manager (NPM)
2) Install Expo: ”npm install -g exp”
3) Download Expo app for iPhone or Android
4) Download Music Theory Application repository
5) Navigate to ”fifths” sub-folder
6) Run ”npm install”

23.3 Running the Project

To get up and running using Expo:

1) Navigate to ”fifths” sub-folder
2) Run ”npm start”
3) Use the Expo app on your phone to scan the QR code, or have the server send you a link.
4) Run the app on your phone.

To use the App:

- Tap or drag to rotate the Circle of Fifths and change keys.
- Tap or drag the sidebar to change or explore relationships between keys.
- Tap the toggles to change between Major/Minor, show/hide relative or parallel keys.
- Tap show chord to show the notes that are in the chord.
- Swipe from the left or click ”Menu” to bring up the menu.
- Tap ”Compose” to bring up the composition page.
- Change the key using the set of radio buttons labeled Key.
- Choose the chord to enter next using the radio buttons labeled Chord.
- Tap ”Add Chord” to add the chosen chord.
- Tap ”New Composition” to start over with a new composition.
- Open the Menu and choose a reference page.
- Swipe up/down to scroll through the reference page.
- Swipe left/right to scroll to the previous/next reference page.

24 Recommended Resources For Learning More

- Lukas Hein, Music Instructor at Western Oregon University
- Perfect Fifth Android: https://play.google.com/store/apps/details?id=com.tritone.perfectfifth
- Perfect Fifth iOS: https://itunes.apple.com/us/app/perfect-fifth/id1382879040?mt=8
- React Native: https://facebook.github.io/react-native/
- React Native Tutorial: https://facebook.github.io/react-native/docs/tutorial.html
- Expo: https://docs.expo.io/versions/latest/

25 Conclusions And Reflections

25.1 Christopher Hebert

I learned very useful technical skills, particularly

- How to create cross-platform apps in React Native
- How to develop and test apps in real time using React Native
- How to use the circle of fifths to compose
- The process of creating a developer account for iOS
- The process of selling/updating an app on the iOS store
- How to use Overleaf

I learned some non-technical things as well. I learned who the best people to communicate with are in a given context. I learned that even when people do strange things sometimes, it doesn’t mean they aren’t honest.

I have learned that project work is actually quite nice if you don’t let your anxieties get the better of you.

I have learned that if you have ambitious ideas, everybody needs to be on board to make them work, especially the client.

I have learned that for teams to function, problems need to be identified and fixed early, and communication needs to be strong and ubiquitous.

If I could do it all over again, I might do it differently. I would buy a used Mac, or use a school computer in order to develop a native iOS app. I might implement some of my own ideas and see how they went over, rather than waiting for permission. I would use Overleaf from the beginning of the term to format latex documents.

25.2 Aidan O'Malley

I learned a lot of very useful technical information over the past three terms while working on Perfect Fifth. Most of my technical education came from using React Native. I had used React in the past and actually was working part-time as a React developer during the school year, however, I only ever used React Native once for a tiny game that I was testing out. There are so many nuances to using React Native that I would've never expected or knew about without working on this project. It has really opened my eyes to the differences between developing for the web and developing for a mobile device. One of the hardest challenges I faced was drawing the circle of fifths onto a mobile device because a similar task is so easy on the web due to there being a canvas. It seemed like similar solutions for a mobile phone were very hacky because the solutions focused more on mimicking what is done in the browser rather than coming up with a mobile specific solution. I also learned a lot about how many bugs crop up during the development of an application like this. Unless you are truly able to think of every possible interaction a user might try to make on the phone, there's no way you're going to be able to find and resolve every bug before releasing an application. This made me really realize that the development lifecycle is not going to end as long as the application is on any application store and as long as you want people to enjoy using the product.

I learned a great deal of non-technical information as well be working with Chris and Lukas so often. I felt as though we had created a startup and each week there was a new task that we had to deal with whether it related to development, design, or some other miscellaneous issue such as intellectual property or splitting up profits from the application. Aside from the business aspects of working on this project, I also learned a ton about music from Lukas. I've begun working on writing more of my own pieces for my guitar and I feel as though that has been one of the biggest benefits I've received from working on this project.

I've learned that project work is all about communication and making sure everyone is on the same page and is happy about the work that is going into the project along with the results of that work.

I've learned a bit about project management in the sense that setting strict deadlines at least gets people to set aside time to work on the project to attempt to meet the deadline. There were plenty of occasions where we wouldn't set deadlines and work would get pushed back, whereas if we set weekly goals, even if they weren't met every week, at least some progress was continually made.

I learned that working in teams is much easier than being alone when developing an application. As long as communication is good and teammates are reasonable, issues can easily be resolved.

If I could do it all over again, the main thing I would've done differently would be to plan out my code for longer before beginning to write it. I think that was one of the biggest hurdles I faced the entire year and it mostly became apparent between Fall and Winter term. I had written the application prototype pretty lazily and didn't plan out much of that code, and instead of starting over from scratch, I tried to just extend on that prototype to create the final version of the Circle of Fifths page. This caused a lot more problems than it solved so in the future I will definitely plan out the structure of each component in a React project and how they will relate to one another before starting to code.

I would also like to get better at testing user interface components with React in the future and that's something I would implement at the very beginning of the development lifecycle if I were to start the project over again. If I had

initially set up tests for all of the key changes for the different user interactions, I would know if I broke anything in the codebase after any change I made by just running all of the tests. Yes, this would have taken a lot more time to implement, but I think it would definitely be worth it and I plan on putting more thought into testing in the future.

26 APPENDIX 1: ESSENTIAL CODE LISTINGS

The following image shows the structure of the CircleOfFifthsScreen (or Home screen) and all the Components that compose it.

```
const CircleOfFifthsScreen = () => (
  <View
    style={{
      flex: 1,
      flexDirection: 'column',
      justifyContent: 'space-between',
    }}
  >
    <View
      style={{
        flex: .8,
        flexDirection: 'row',
      }}
    >
      <View style={{ flex:1 }} />
      <View
        style={{
          flex: 4,
          flexDirection: 'column',
          justifyContent: 'space-between',
        }}
      >
        <View
          style={{
            flex: 1,
          }}
        >
          <Toggles />
        </View>
        <View
          style={{
            flexDirection: 'row',
            flex: .1,
          }}
        >
          <View
            style={{
              flex: .13,
            }}
          >
          </View>
          <View
            style={{
              alignItems: 'center',
              flex: .75,
            }}
          >
            <Text style={{ fontSize: 20 }}>↓↙</Text>
          </View>
        </View>
      </View>
      <View
        style={{
          flex: 1,
        }}
      >
        <Sidebar />
      </View>
    </View>
    <View
      style={{
        flex: 1,
      }}
    >
      <CofContainer />
    </View>
  </View>
```

The following image shows the COFContainer component's structure and the components that compose it.

```
<View style={styles.container}>
  <View { ... this._panResponder.panHandlers}>
    <Surface width={Dimensions.get('window').width} height={Dimensions.get('window').height}>
      <Group x={x} y={y} transform={new Transform().rotate(this.state.rotation)}>
        { /* full circle of colored wedges */ }
        <Circle
          radius={radius}
          innerRadius={radius / 10}
          colors={colorArraySelect(this.props.currentScale)}
        />
        { /* outer black ring */ }
        <Circle
          radius={radius * (14 / 18)}
          innerRadius={radius * (14 / 18) - .25}
          colors={new Array(12).fill('#2a2a2a')}
        />
        { /* inner black ring */ }
        <Circle
          radius={radius / 2}
          innerRadius={radius / 2 - .25}
          colors={new Array(12).fill('#2a2a2a')}
        />
        { /* Circular text for current key */}
        <CircularText
          data={this.props.currentScale === 'maj' ?
            ['#4', 'vii°', 'iii', 'vi', 'ii', 'V7', 'I', 'IV', '', '', '', '♭2'] :
            ['#4', '', '', '', 'ii°', 'V7', 'i', 'iv', '♭VII', '♭III', '♭VI', '♭2']
          }
          rotation={-this.state.rotation}
          colors={new Array(12).fill('#2a2a2a')}
          multiplier={1.32}
        />
        { /* Circular text for all notes */}
        <CircularText
          data={this.props.fifths}
          rotation={-this.state.rotation}
          colors={new Array(12).fill('#2a2a2a')}
          multiplier={1.12}
        />
        { /* Circular text for parallel notes */}
        {this.props.showParallel ?
          <CircularText
            data={this.props.currentScale === 'maj' ?
              ['', 'vii°', 'iii', 'vi', 'ii', 'V7', 'I', 'IV', '', '', '', ''] :
              ['', '', '', '', 'ii°', 'V7', 'i', 'iv', '♭VII', '♭III', '♭VI', '']
            }
            rotation={-this.state.rotation}
            colors={new Array(12).fill('#2a2a2a')}
            multiplier={0.6}
          /> : null
        }
        { /* Circular text for relative notes */}
        {this.props.showRelative ?
          <CircularText
            data={this.props.currentScale === 'maj' ?
              ['', '', '', '', 'vii°', 'iii', 'vi', 'ii', 'V7', 'I', 'IV', ''] :
              ['', 'ii°', 'V7', 'i', 'iv', '♭VII', '♭III', '♭VI', '', '', '', '',]
            }
            rotation={-this.state.rotation}
            colors={new Array(12).fill('#2a2a2a')}
            multiplier={1.82}
          /> : null
        }
      </Group>
    </Surface>
  </View>
```

The following image shows the code for handling which notes make up the 12 fifth intervals in the Circle Of Fifths and the code that determines what the current rotation angle should be.

```
import { createSelector } from 'reselect';
import { majorQualities, minorQualities, tonalGravity } from '../static/keySignatures';

// gets the 12 intervals of five related to the current key
export const fifthsGenerator = (key) => {
  // get root note's index
  const index = tonalGravity.findIndex(el => el.note === key);
  // gets the 12 notes that make up the fifth intervals starting at this root
  const notes = tonalGravity.filter((el, i) => {
    // these are the notes that are actually in this key/scale combo
    if(i >= index - 6 && i <= index + 5) {
      return el;
    } else {
      return null;
    }
  }).map(el => el.note);
  return notes;
}

// selector that will update component's that the selector is referenced in
// whenever the inputs to this function change
export const fifths = createSelector(
  state => state.keys.currentKey,
  (key) => fifthsGenerator(key),
);

// selector that will update component's that the selector is referenced in
// whenever the inputs to this function change
// returns the correct rotation angle to put the new root to the top
export const rotation = createSelector(
  state => state.keys.currentKey,
  state => fifths(state),
  (currentKey, fifthNotes) => {
    // multiplier is essentially the number of wedges away from the first wedge we currently are
    const multiplier = fifthNotes.length - 1 - fifthNotes.findIndex(el => el === currentKey);
    // get the angle we need to rotate to
    const angle = (multiplier * 30) + 15;
    return angle;
  },
);
```

Below code shows how tonal gravity is analyzed. It is a function of the positions in the circle of fifths of two adjacent chords:

```
export const tonal_gravity_transition = function(fifth_pos1, fifth_pos2) {
    // one of [lu, ld, su, sd, pm]
    // Calculate the transition from r1 to r2
    var f1 = fifth_pos1, f2 = fifth_pos2;
    if (f1 === f2) {
        return parallel_motion;
    }
    if (f2 === f1 - 1) {
        return step_down;
    }
    if (f2 === f1 + 1) {
        return step_up;
    }
    if (f2 < f1) {
        return leap_down;
    }
    return leap_up;
};
```

The following code shows how an individual chord is analyzed for substitutions:

```
const composition_chord_analysis = function(chord, next_chord, key_chords, parallel_key_chords) {
    // Analyze a chord as part of a composition_analysis

    // Analyze diatonic chords
    if (position_chord(chord, key_chords) !== null)
        return make_analysis(chord, fifth_position(chord.root), diatonic, next_chord);

    // Analyze borrowed chords
    if (position_chord(chord, parallel_key_chords) !== null)
        return make_analysis(chord, fifth_position(chord.root), borrowed, next_chord);

    // Analyze secondary dominants
    if (chord.quality === dominant &&
        next_chord &&
        position_chord(next_chord, key_chords) &&
        fifth_position(chord.root) - 1 === fifth_position(next_chord.root)) {

        return make_analysis(chord, fifth_position(chord.root), secondary_dominant, next_chord);
    }

    // Analyze tritone substitutes
    if (chord.quality === dominant &&
        next_chord &&
        (fifth_position(chord.root) + 5 === fifth_position(next_chord.root) ||
         fifth_position(chord.root) - 7 === fifth_position(next_chord.root))) {

        return make_analysis(chord, fifth_position(next_chord.root) + 1, tritone, next_chord);
    }

    // NOTE: Do not try to analyze diatonic substitutes

    // Unknown substitution
    return make_analysis(chord, fifth_position(chord.root), unknown, next_chord);
};
```

Finally, the "composition_analysis" function analyzes the entire composition for substitutions:

```
// Perform a full analysis of chords in a composition, in the given key.
export const composition_analysis = function(composition_chords, key) {
    // Analyse the composition_chords
    var key_chords = key_diatonic_chords(key), parallel_key_chords = key_diatonic_chords(parallel_key(key));
    var analysis = [];

    for (var i = 0; i < composition_chords.length; ++i) {
        var chord = composition_chords[i];
        var next_chord = i !== composition_chords.length-1 ? composition_chords[i + 1] : null;
        var tmp = composition_chord_analysis(chord, next_chord, key_chords, parallel_key_chords);
        analysis.push(tmp);
    }
    return analysis;
};
```