DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Audio Visualisation – Approaching a Music Improvisation Learning Process

Benedikt Wimmer

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Audio Visualisation – Approaching a Music Improvisation Learning Process

Audiovisualisierung: Unterstützung eines Lernprozesses zur musikalischen Improvisation

Author:	Benedikt Wimmer
Supervisor:	Prof. Dr. Rüdiger Westermann
Advisor:	M.Sc. Christian Reinbold
Submission Date:	15/9/2019

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15/9/2019 Benedikt Wimmer

Abstract

In order to implement a new software approach for guided musical improvisation, we investigated how jazz music improvisation is taught to a broad audience today. This revealed that popular solutions mainly rely on standard music notation. As it can be argued that standard music notation is not ideal for jazz music and is not intuitively readable for users without appropriate training, our goal is to provide a new alternative to standard music notation for jazz music. We propose a novel notational concept that is based on the well-known idea of displaying musical information on a piano keyboard. We describe its properties and report on the implementation of a software prototype that combines a range of possible designs for the new notation with features of recent software solutions in the field of music visualisation. The prototype was evaluated in a user study.

Contents

1. Introduction

Jazz is a musical discipline that heavily depends on improvised elements. As improvisation is an art form inherently dependent on interaction and chance, there is supposedly no necessity to overanalyse it or put it into rigid schemes. However, in order to propagate knowledge about jazz music nonverbally, it was a natural consequence that jazz musicians have started to search for suitable ways to formalize information about their style of music. Using musical *staff* notation as shown in Figure 1.2 would be too specific and detailed for the needs of jazz musicians, as sheet music notation depicts what notes *must* be played rather than what *can* can be played by a musician in order to correctly interpret it. So instead of depicting music as specific notes, it was desirable to find a format to outline the musical frame without limiting players too much in their creative freedom. During the mid of the 20th century, this ultimately led to a musical format called *lead sheet* notation and was later used to publish songbooks like the *RealBook* [8]. Today, lead sheets as depicted in Figure 1.1 are a standard and known by jazz musician accross the world and a lead sheet is usually the only music-theoretical information players have at hand while improvising.

1.1. Motivation - Overcoming the Difficulties of Lead Sheet Notation

That means that, essentially, they can see what kinds of chords are played at a specific point in a song, but it also means that they have to break down sequences of chord symbols to possible musical scales and melodic motives on their own (see Chapter 3). As, for some songs, chords can mutate in fractions of seconds, the chord symbol information has to be broken down very fast while simultaneously keeping track of the harmonic progression. This is mandatory as in order to find out the correct scale, one must first make sure that they are always aware of the current chord [28, p. 256]. Furthermore, an improviser needs to make choices about parameters like their articulation, use of rhythmic and melodic patterns as well as handle instrument-specific technical challenges. It is fair to say that, in musical improvising, a musician's brain is especially tested in its ability to multitask. While none of the abovementioned tasks is too challenging for the human to be processed in real time, the combination of them is

what adds up to make improvisation a hard skill to learn and master.

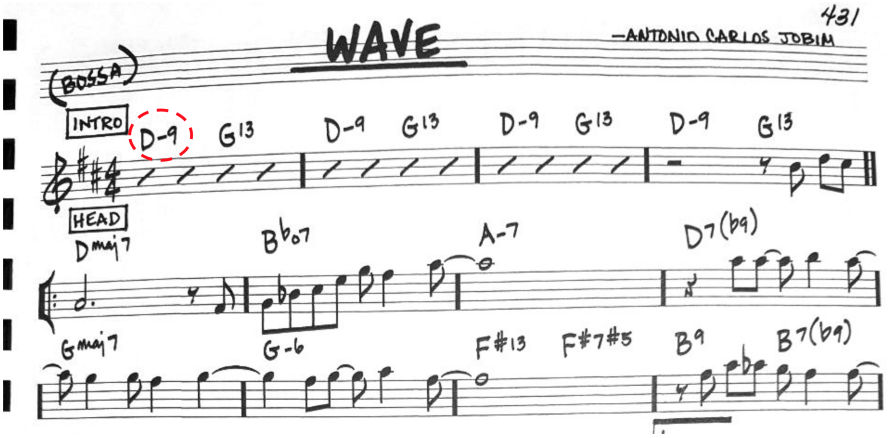

Figure 1.1.: A lead sheet for the song *Wave* from *The Real Book* [8, p. 431]: Lead sheets combine chord notation with melodic notation. In the *INTRO* section, one can observe pure *chord chart* notation, in the *HEAD* section, chord chart notation coexists with melodic notation. The first played chord in this song is highlighted in red.

Looking at improvisation as a "sum of tasks", it is always a good idea to eliminate one or some of those tasks in order to reduce complexity. While there are methods as described in Chapter 2 that already achieve this goal to a certain extent, there is still one problem for the player which seems to be neglected: The lead sheet notation. As described above, it is up to the player to interpret the given chord information, which requires advanced music theory knowledge (see Chapter 3). Obviously, replacing lead sheet notation with another type of notation where chords are pre-translated to full scales, could mean one task less to a music learner. In fact, in *iRealPro* this is solved by optionally displaying scales notated in staff notation for the current chord (see Chapter 2.2). Whilst being a valid approach, this yields the disadvantage of introducing standard music notation. As reading sheet music is not necessarily a skill known to every musician, sheet music might add a new kind of complexity to the learning process. Furthermore, one can argue that sheet music does not always display according to what a player is hearing (see Figure 1.2).

This work aims towards providing a new, but intuitively readable visual representation for suggested chord/scale pairs together with other minor features.

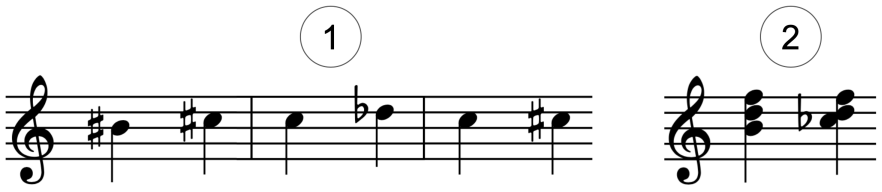

Figure 1.2.: Ambiguity in sheet music: In *1*, one can see the very same melodic phrase notated in three different ways. Note that the third version does not vertically display an ascend of pitch (both notes share the same natural). In *2*, one can see the same ambiguity for a diminished chord.

1.2. An Application as a Referenceable and Mutable Medium

Aiming towards a proof of concept, our visual representation will be embedded into a real time application. This yields the possibility of exploiting all advantages of software like *iRealPro* and additionally implementing new features, thus giving an opportunity to compare the new solution with already existent ones. For our application, though being very useful and reasonable features, not all of the features from *iRealPro* (see 2.2) will be provided. For the sake of conceptual comparison, it suffices to have the possibility of adding them at a later point in time. Those missed-out features include time-stretching, a song database, song creation/editing by the user, transposition and computer-created backing tracks. After taking a short look at the remaining features of *iRealPro*, it is possible to think that our main application of being guided through a leadsheet musically and visually could as well be accomplished by creating videos featuring our new visual chord representation, but this would mean missing out on several future runtime options only a software solution can provide.

As mentioned above, there are additional new features we want to implement. In contrast to related work like *Aebersold*, *Musiclock* and *iRealPro* (see Chapter 2), we make it a priority to give the user feedback on their current pitch. While for some instruments, like the piano, this is certainly not necessary, vocalists and wind instrument players often struggle to know the exact notes they are singing or playing and are therefore insecure about their melodic context. For those players, inner imagination of pitch is crucial to their improvisation performance and could be enforced by accurate pitch feedback like we can see in *Vocal Pitch Monitor* or a conventional tuner. To anticipate, this design choice mandatorily introduces a real time constraint. Chapter 4 describes, in which way this guided improvisation experience is realized featuring a new chord/scale representation and real time pitch feedback.

2. Related Work

In this section, different media-supported approaches in music pedagogy are presented and discussed regarding their methodology, benefits and possibilities.

2.1. Jamey Aebersold

Jamey Aebersold is an US-American jazz educator. In 1967, he began releasing a series of playalong books, which have since become a popular resource for practicing jazz improvisation [10]. In those books, the reader can see sheet music for jazz standards as well as suggested scale choices for those song's respective chord sequences 2.1. They come with backing tracks for each song, so the user is able to practice on their own while having the experience of playing within a band.

The scale suggestions can help the reader work on their soloing without a need for them to prepare own scale choices before or while improvising. However, being able to read sheet music is mandatory for using this feature and using Jamey Aebersold's books in general (see Chapter 1.1). Furthermore, musicians using these book have to be able to follow the time-flow of a chord progression on their own.

2.2. iReal Pro

iReal Pro is a cross-platform application dedicated to the domain of jazz music and developed by *Technimo* [26]. Essentially, the vast majority of known jazz standards is stored online and is accessible to the user. After downloading a song, the user can play to a computer-generated backing track. In its standard mode, the app does only display a chord chart for the user to draw harmonic information from.

The dynamic nature of software does have some advantages over books or written word. For example, while the *Aebersold* series had to be published in several versions transposing instrument groups, in *iReal Pro* this can be adjusted on the spot without any changes to the stored data. Moreover, parameters like tempo and musical style can easily be adjusted to the user's wishes and users are able to edit/create songs. This is realized by providing a virtual band to accompany the user while playing. This band consists of a piano or guitar, a bass and drums, which create chords, melodies

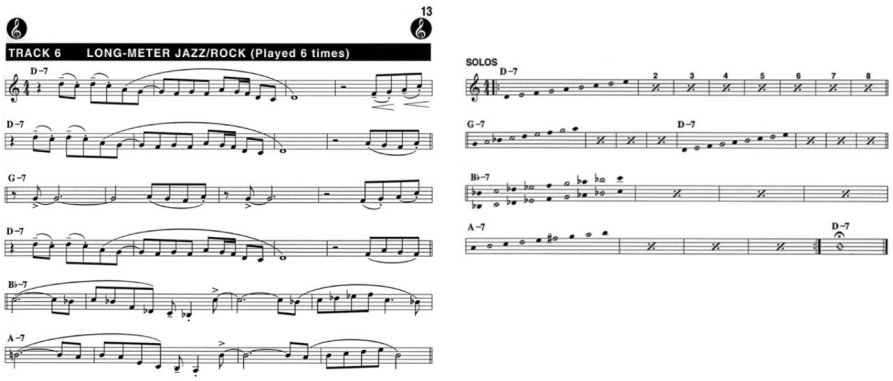

Figure 2.1.: An excerpt from *Aebersold Vol.2 Nothin' But Blues* [1, p. 13]: On the left one can see chords and melody in lead sheet notation. On the right one can see chords being represented together with a suggested scale. Note that chord notes are displayed as filled dots while scale notes are displayed as circles.

and rhythms through pre-defined, pseudo-random patterns which are adjusted to the musical style and tempo the user desires. Following the chords' progression and skills in sheet music reading are no longer required, as the current bar is always highlighted and no melodic information is displayed here (see Figure 2.2).

This brings us to the first disadvantage of this app compared to *Aebersold*. By Design, it does not display the melodic theme of a standard. A second downside lies in the display of scale choices. *iReal Pro* can optionally suggest a scale for the current chord, but these suggestions are computer-generated, while, in *Aebersold* books, these are picked by expert jazz players. *iReal Pro*'s suggestions for chord-scale pairs are generic to an extent, where, for example, for a *D minor* chord the suggested scale is always the *D dorian* scale. In reality, there would be more options like the *melodic minor, harmonic minor, lydian and phrygian* scale and – depending on the context – *dorian* might be musically incorrect to choose (see 3), a fact neglected by *iReal Pro*.

2.3. Synthesia

Synthesia is an application designed by *Synthesia LLC* [24]. In contrast to *Aebersold* and *iReal Pro*, *Synthesia* is not constricted to the domain of jazz and is a tool for learning the piano. The pianist develops their skills by imitating visual patterns displayed by *Synthesia* with their own hands and a keyboard plugged to the computer. As can be expected from a software solution, the program offers the option to adjust tempo and transposition. Here, the abovementioned patterns serve as a replacement

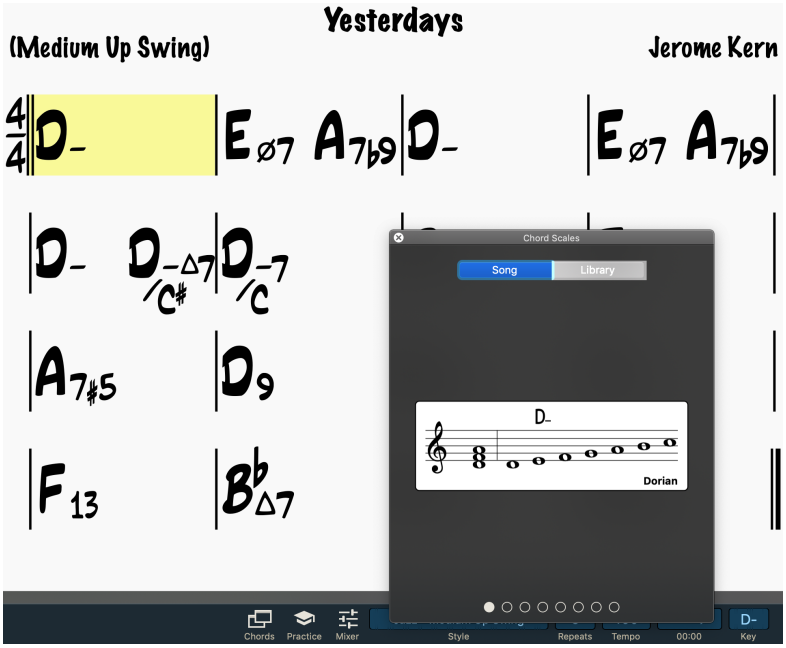

Figure 2.2.: A screenshot of *iReal Pro* [26]: *D minor* is highlighted as the current chord and scale suggestions are displayed.

for traditional music notation. They are created from *MIDI* (a standard to formalize musical information so that it can be accessed by digital devices [17]) files and displayed as bars with a certain length, which corresponds to duration, and a certain position. This position can be interpreted vertically and horizontally as time and pitch (see Figure 2.3). This representation can be interpreted by players of all skill levels without a need to read from sheet. Such a representation is also widely known as a *piano roll*.

2.4. MusiClock

MusiClock offers several products. For this thesis we want to focus on their mobile application [20]. The application aims towards an user experience where the user melodically improvises to a backing track, guided in such a way that the application displays a correct scale choice and the user can pick from the suggested notes. While featuring scale representations on piano, guitar, and sheet music notation, it introduces a new concept where all twelve musical notes are displayed on a circle (see Figure 2.4). This is done clockwise beginning from *C*. To represent scales, there is a template on top of that circle, which is "cut out" such that it only displays the suggested notes

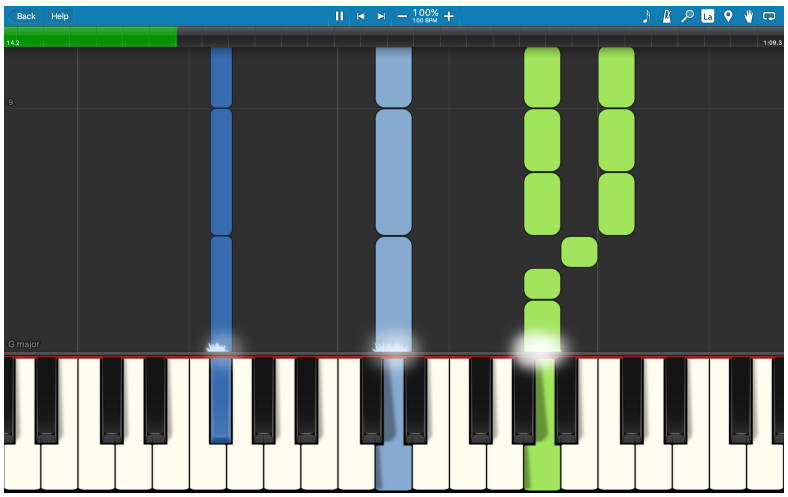

Figure 2.3.: A Screenshot of *Synthesia* [24]: *MIDI* information is represented horizontally and vertically relative to a piano and the song time. The piano keys are highlighted according to the player's input.

and hides all others. Templates for different scale types have different shapes, but for transposition, one can use the same template and rotate it. This means, that e.g. *D mixolydian* and *Ab mixolydian* share the same, but rotated template.

This notation's advantage is that the user is neither dependent on knowledge about sheet music notation nor about a specific instrument. Additionally, it is able to show transposition without changing a template's proportions. This helps the user develop a correct imagination of music-theoretical structures.

In the *MusiClock* app, once a scale is chosen, the musician plays over the backing track while *MusiClock* shows that particular scale. The scale cannot be switched while a track is playing, meaning that there is currently no possibility to account for more complex harmonic progressions, which, to be interpreted correctly, would mandatorily need the improviser to switch scales at some point (see 3). This observation makes it evident that *MusiClock* is unable to display changes of musical context.

2.5. Vocal Pitch Monitor

Vocal Pitch Monitor is a mobile application by Tadao Yamaoka [29]. It is designed to give a singer or instrumentalist visual feedback about the pitches they are producing. Therefore, a graph showing a pitch history is drawn in real time. The pitches are displayed in such a way that all twelve notes are vertically distributed in an equidistant

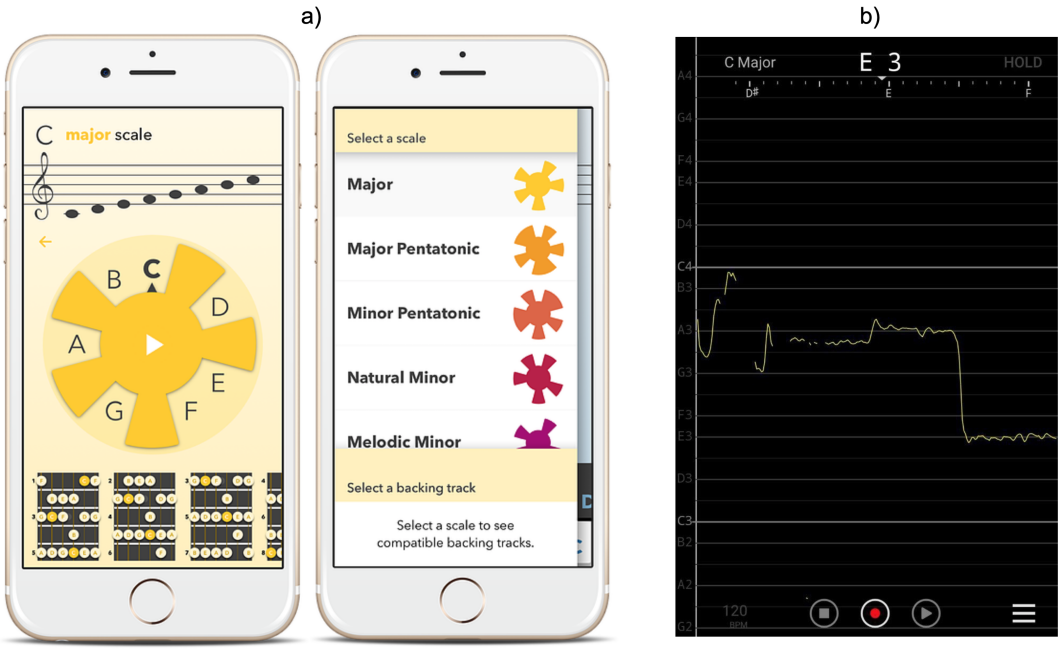

Figure 2.4.: a) Examples from the *MusiClock* website [20]: On the left, one can see a *C major* scale represented on a staff and a guitar fretboard as well as *MusiClock*'s main representation with a template hiding irrelevant notes. Rotating the tablet to the left by one step would cause a *B major* scale to be displayed. On the right, one can see templates for several scales, from which the user can choose before being presented with matching backing tracks. **b)** A screenshot of *Vocal Pitch Monitor* [29]: The screen center displays a pitch history graph with the rightmost area showing the present pitch. Here, the graph is displayed relative to a *C major* scale (white horizontal lines). In the top area, the note name of the current pitch is displayed together with its octave.

order (see Figure 2.4).

Like *MusiClock*, this equitable display of the twelve notes helps the musician develop a well-balanced imagination of the relation between pitches and note names. Here, the user's physical experience of pitch is supported even further by the visual perception of the vertically aligned pitch history.

2.6. Possibilities for a new application

The advantages of real time software in guiding users through a flow of musical information over other media was formulated especially in Chapter 2.2. We remember that software is able to do tasks like following time, transposition and editing song information dynamically. Even though *iReal Pro* provides such functionality, there is still a huge challenge for the user in overcoming the barrier of musical notation (see Chapter 1.1) and chord symbols (see Chapter 3) in order to be able to improvise over the musical information provided by *iReal Pro*. The same statement holds true for Jamey Aebersold's book series. However, it would indeed be possible to substitute the standard musical notation used in *iReal Pro* and Jamey Aebersold's work with other approaches like the keyboard presentation in *Synthesia* or *MusiClock*'s templates.

This is why we have chosen to propose a new approach to visual guidance of music improvisation that aims to be more friendly towards users with less musical background. In that, we want to keep the mentioned advantages a software solution like *iReal Pro* provides, but with a more intuitively understandable visualisation of chord/scale information. *MusiClock* provides a very clear approach towards scale visualisation, but – while intuitively comprehensible – its visualisation method is unfamiliar even to users with musical background. Furthermore, this method uses two – horizontal and vertical – visual dimensions to display its pitch information on the circle. *Synthesia* uses a piano keyboard as a grid for musical pitch, thus displaying musical pitch in only one – horizontal – dimension. This opens up the possiblity of easily displaying a flow in time on the vertical axis. Displaying a flow in time together with pitch appears to be less natural to a *MusiClock*-like visualisation. It would be possible to add a third visual dimension or place several *clocks* next to each other, but it would not yield the same beauty by design as *Synthesia*'s time display. Furthermore, as opposed to *MusiClock*'s represenation, a keyboard visualisation is more likely to be familiar to even a user without musical background. According to the previous argumentation we decided to orient our visual representation of musical information towards a piano roll.

To make a *Synthesia*-like experience of imitating "falling notes" accessible not only to users who own or play a keyboard, it was chosen to go down the path of pitch dection as featured in *Vocal Pitch Monitor*. With this feature it is possible to display pitches produced by all kinds of musical instruments on a piano keyboard. The exact implementation of pitch visualisation on a piano keyboard as well as the interaction between chord visualisation, pitch visualisation and the piano roll layout is specified in Chapter 4.

3. Background in Music Theory

In the following chapter, basic music theoretical concepts for understanding the incentive of this thesis will be explained to the reader. It is assumed that the reader is knowledgeable about *intervals*, the *Major* and *Minor mode*, *stressed* and *unstressed beats*, and standard music notation as well as the note names on a musical keyboard. Note that this work is not about music theory but about music visualisation, therefore this chapter will not take into account stylistic specifications of jazz music. For further reading, Mark Levine's *The Jazz Theory Book* [16] is recommended, though for speakers of german language, Frank Sikora's *Neue Jazz-Harmonielehre* [22] is the preferred recommendation.

3.1. Transposition

One of music's fundamental concept is *transposition*. In short, it says that every musical scale, melody, chord, etc. or a combination of those constructs can be displaced from its original pitch to an arbitrarily different pitch, as long as its inner proportions stay the same. This is demonstrated in Figure 3.1. As a note, those mentioned inner proportions of musical constructs can be broken down to a set of pairwise intervallic relations between the notes involved in such a construct. Those intervals are defined as fixed frequency ratios between two notes, such that the transposition of an entire musical construct basically means multiplying the frequencys of all of its involved notes by a certain constant [18, p. 117]. For every concept explained in this chapter, transposition can be applied and therefore these concepts are universally applicable for all possible pitches.

3.2. Interdepencence between Melodies, Chords and Scales

Musical *scales* consist of a certain number of musical notes which inherently have a certain proportion to each other. They could be mathematically described as a set of notes. One common and therefore exemplary scale is the *major* scale (see Figure 3.2).

Many songs in popular and classical music are based on a certain underlying scale. All notes played in such songs can be found in the song's underlying scale's set of

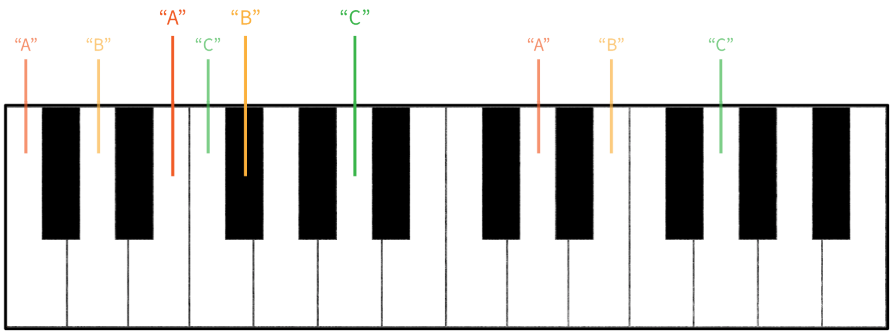

Figure 3.1.: The first three notes to the chorus of *The Jackson Five*[s'] single *ABC* [27] famously represent the first three entries in the english alphabet. In *The Jackson Five*'s version, these notes' note names are *E*, *F#* and *A*. However, when keeping the pitch proportions between those three notes the same, one can transpose the melody to a lower or higher pitch without a loss in recognisability.

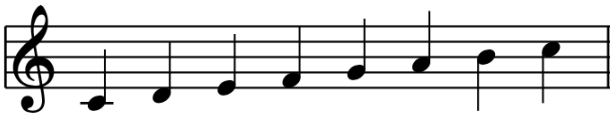

Figure 3.2.: The *C Major* scale notated in *treble clef*. It consists of seven distinguishable notes, which is common for western music (see [22, p. 42]) and can be observed taking into account that lowest and highest notes depicted are both a *C*.

notes (see Figure 3.3).

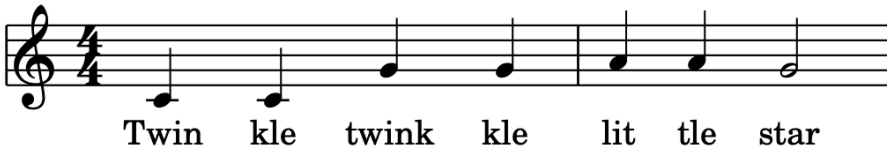

Figure 3.3.: The melody from the beginning of Jane Taylor's *Twinkle, twinkle, little star* [25]. The song's underlying scale is *C Major*. A comparison with Figure 3.2 confirms that all notes from this song are included in the *C Major* scale.

From each note or *position* in a scale, it is possible to form a chord. Chords can be formed by stacking *thirds* from the respective scale note (see Figure 3.4).

The higher a stack of thirds, the more complex a chord will sound. Chords formed from two stacked thirds are often used to accompany melodies in children's songs or

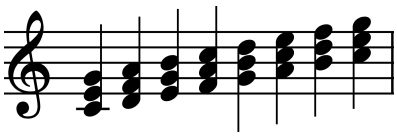

Figure 3.4.: Chord constructions using two stacked thirds on all positions in the *C Major* scale.

classical music like Mozart's *Eine kleine Nachtmusik* [13, p. 414], while the use of three or more stacked thirds is characteristic for jazz or jazz-related music [22, p. 67]. Different combinations of *minor* and *major thirds* determine the chord's inner proportions, which is referred to as *chord type*.

As we have found (see Figure 3.4), chords consist of a set of notes. If for example a pianist wanted to use chords to accompany a melody, a common approach is to make sure that their chord choice is built within the melody's scale and contains the current melody note, especially if the note is played on a stressed beat. Often, common accompaniment solutions for a song are annotated to songs using *chord symbols* (see Figure 3.5).

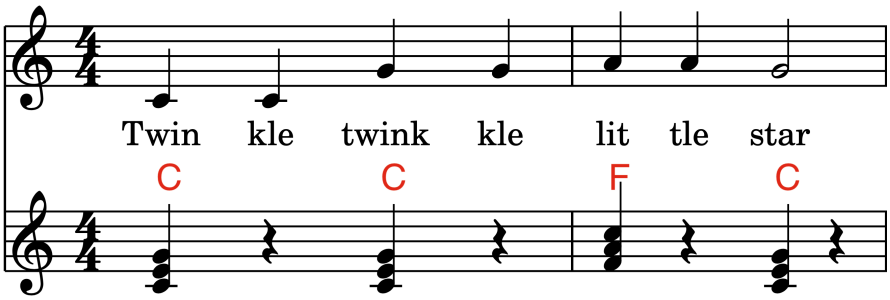

Figure 3.5.: A possible accompaniment to the melody of *Twinkle, twinkle, little star*. Note how each chord in the accompanying instrument contains the rhythmically respective melody note. Also note that all the chords were chosen from a *C Major* context (compare 3.4) and can be represented as chord symbols (red).

We have now looked at an approach of creating music where chords are picked for an already existing melody. This unidirectional approach to music creation is certainly not standard for most kinds of music, in fact it may as well work the opposite way round with melodies being composed to fit existing chord progressions. This is the case with melodic improvisation, as – for the improviser – the challenge here is to find melodies to given chords (see Figure 3.6). In contrast to before, where an accompanying instrumentalist had to find matching chords using the melody's scale, the melody now

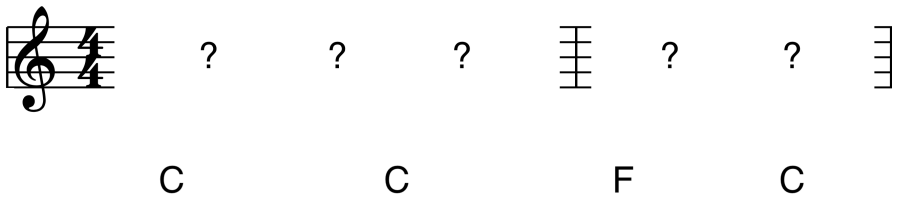

Figure 3.6.: Flipping the script: An improviser has to find a possible melody to a given chord progression. The goal is now to find a melody, which – on the current beat – consists of notes that are contained in the given chord.

has to consist of matching notes using the chord progression's underlying scale. For a melody player, using the whole of a scale opens up the possibility for more melodic variety and harmonic tension [22, p. 89f.] as opposed to using only notes included in the given chord, but guessing the correct scale for melodic improvisation can lead to ambiguous results. Observing that the chords in the example (see Figure 3.6 and explicit notation for the chord symbols in Figure 3.5) determine six distinguishable notes and most western music scales being defined by a set of seven notes (see [22, p. 42]), several possible scales come into question. Here, *C Major* can be determined as the correct underlying scale given that the melody to *Twinkle, twinkle, little star* starts and ends on a *C*.

3.3. Alternating Sequences and Chord-Scale Theory

There are cases where – unlike for *Twinkle, twinkle, little star* – chords and melody cannot be constructed using only one underlying scale. During a song, the tonal material can change in such a way that the underlying scale has to be mutated or *altered* in order to include the new tonal material. This alteration ideally results in a new scale by its own right. However, the way in which this alteration happens can be ambiguous, in which case the player has to make a decision based on further context. See Figure 3.7 for an example of an alteration in Gerald Marks and Seymour Simon's song *All of Me* [8, p. 16].

Because of how common tonal alterations are in jazz music, it is common practice to re-determine the current scale for each new chord symbol. In order to execute this approach, musicians use the concept of *chord/scale theory* [16, p. 29-88]. A chord's type is used to predetermine a set of possible scale types for this particular chord. Remembering that chords are formed using their underlying scale (Figure 3.4), one can conclude that therefore a valid scale choice must include all notes played in the respective chord. A resource for possible chord/scale pairs is Jamey Aebersold's

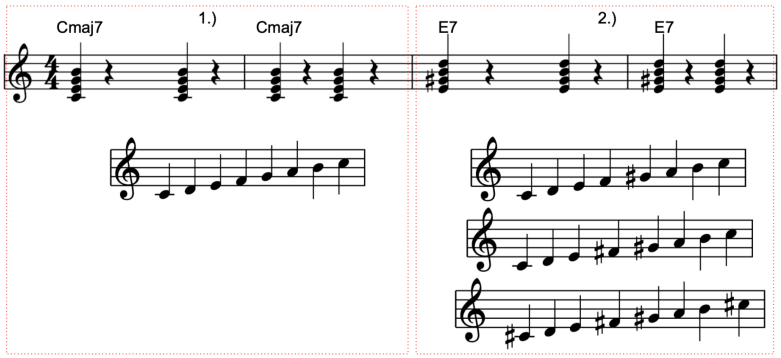

Figure 3.7.: The beginning of *All of Me*: **1.)**: The first two measures can be interpreted using a *C Major* scale, as this is the song's main mode. **2.)**: A *G#* is introduced through an *E7* chord, which is not part of *C Major* and thus the scale has to be altered. Three possible options for a sufficient alteration are depicted. They all include the notes *E*, *G#*, *H* and *D* from the *E7* chord. While the first option is the closest to *C Major*, it would make sense to opt for the second or third one if an *F#* or even a *C#* are introduced during the following chords, which are not depicted here.

Introduction to the Scale Syllabus [9]. Having a set of scales at hand for each chord, it is now up to the musician to make scale choices to best represent a musically cohesive interpretation of a given chord progression. We can now have a detailed look at how chord/scale theory can be applied to a given chord progression of 4 chords:

D7 G-7 C7 F△

An exemplary tabular based on *Introduction to the Scale Syllabus* [9] gives us possible scales for each required chord type.

C△	**C7**	**C-7**
C Major	*C Mixolydian*	*C Dorian*
C Lydian	*C Spanish*	*C Phrygian*
C Harmonic Major	*C Lydian Dominant*	*C Aeolian*

For each chord in the given progression, we determine a set of possible scales using transposition (see Chapter 3.1).

D7	**G-7**	**C7**	**F△**
D Mixolydian	*G Dorian*	*C Mixolydian*	*F Major*
D Spanish	*G Phrygian*	*C Spanish*	*F Lydian*
D Lydian Dominant	*G Aeolian*	*C Lydian Dominant*	*F Harmonic Major*

An experienced musician can deduce, that the progression at hand is a variation of a common musical sequence called a *turnaround* [22, p. 224] and thus the F△ chord being the central chord of this turnaround should be interpreted with the scale of *F Major*. Under this assumption, one can proceed to choose scales for the previous chords. As stated in Chapter 3.2, songs from popular music are often based on one single underlying scale. Assuming that this is the style of music most listener's ears are used to perceiving, it makes sense to maximize the scale intersection and therefore minimize scale alterations between a chord and its context chords. In order to do this, we have to look at the concrete notes in each of our scale options.

D7	**G-7**	**C7**	**F△**
D E F# G A B C	G A Bb C D E F	C D E F G A Bb	F G A Bb C D E
D Eb F# G A Bb C	G Ab Bb C D Eb F	C Db E F G Ab Bb	F G A B C D E
D E F# G# A B C	G A Bb C D Eb F	C D E F# G A Bb	F G A Bb C Db E

Applying the assumption that an *F Major* scale is chosen for F△, we will deduce scale choices from right to left. In the following table, notes included in the previous scale choice are written in **bold**, scale choices which yield the biggest intersection with the previous scale are highlighted.

D7	**G-7**	**C7**	**F△**
D E F# G A B C	**G A Bb C D E F**	**C D E F G A Bb**	F G A Bb C D E
D Eb F# **G A Bb C**	**G** Ab **Bb C D** Eb **F**	**C** Db **E F G** Ab **Bb**	F G A B C D E
D E F# G# **A** B **C**	**G A Bb C D** Eb **F**	**C D E** F# **G A Bb**	F G A Bb C Db E

Choosing a *C Mixolydian* and subsequently a *G Dorian* scale for the C7 and G-7 chord can be considered the best solution for our approach, as those two scales consist of the same set of notes as the *F Major*. For D7, the *Mixolydian* and *Spanish* scale both yield 2 alterations. Deciding between those two scales is a matter of taste for the musician, as both scales have different inner proportions and therefore have different musical characteristics.

The chord/scale theory is only one approach to this problem and in chord/scale theory it is certainly possible to choose a different approach than our proposed set intersection metric, e.g. using intuition. As we can see, choosing musical scales for chords is a task that requires not only analytical skill, but also music-theoretical (e.g. determining tonal centers) and practical experience (e.g. distinguishing the characteristics of the *Spanish* and *Mixolydian* scale).

3.4. Guide Tone Lines in Jazz Improvisation

Guide Tone Line is a broad term for melodic lines that manage to outline the harmonic progression while introducing only small melodic movement [22, p. 228]. While guide tone lines can be produced with movement in all possible directions and using all possible intervals of a scale, it is common to practice such melodic lines especially using the *thirds* and *sevenths* of the given chords/scales. To understand why this is done, we have to take a look at common harmonic patterns in jazz music. As can be seen in Eremenko, Demirel, Bozkurt and Serra's *JAAH: Audio-aligned jazz harmony dataset* [6], *falling fifths* are the most common harmonic movement in Jazz. Fortunately, our previous example is – as well as a so-called *turnaround* – a sequence of falling fifths. This is the case as F is a *fifth* below C, which is a *fifth* below G, which is a *fifth* below D. In the following tabular, you can see guide tone lines – represented as arrows – alternating between the *thirds* and *sevenths* of the example's chords. Note that, due to the fact that this is a sequence of falling fifths, the **5** of a chord is the same as the **1** of its left neighbour.

	D7	**G-7**	**C7**	**F△**
1	D	G	C	F
3	F#	Bb	E	A
5	A	D	G	C
7	C	F	Bb	E

As you can see, both of these lines, while outlining the chords step by step, either stay on the same pitch or fall only by a *minor* or *major second* per step. The criterion for a guide tone line is fulfilled. This principle can be applied to every sequence of falling fifths and is therefore applicable to the majority of jazz standards [6], making it an effective skill to practice.

One could achieve similar melodic lines by alternating between the **1** and **5** of each chord, but this would sound less engaging to the listener. To explain why, this can be tried by taking a look at the *JAAH* dataset [6] again. The data suggests that not only the majority of all chords in the data, but also the majority of chords involved in movements of falling fifths have either a *Major*, *Minor* or *Dominant* chord type. A comparison of those chord types as 4-note chords reveals that they differ only in their *thirds* and *sevenths*:

	C-7 (*Minor*)	**C7** (*Dominant*)	**C△** (*Major*)
1	C	C	C
3	Eb	E	E
5	G	G	G
7	Bb	Bb	B

Using this fact and the data, one can reason that guide tone lines alternating between a chord's **3** and **7** yield more discriminative information about the chords played than lines alternating between the **1** and **5**, thus sounding more interesting. Furthermore, from a sound-sensational point of view, it can be argued that there is a starker contrast between the former pair, as here, *thirds* are perceived as *consonant* and *sevenths* are *dissonant*, while *unsions* and *fifths* are both classified as *consonant* [22, p. 24]. This contrast may lead to a more engaging listening experience as well.

4. Designing an Application

As Chapter 1.2 explains, a software application is considered a suitable solution for providing guidance to users exercising in melodic improvisation. To design this application, it was important to start with a list of required attributes. Hence, several instances of related work (see Chapter 2) were assessed with regard to beneficial visual features provided by them. These features cover guidance for musical parameters like time (including tempo) and pitch as well as musical constructs like chords and harmonic progression. In order to provide a state-of-the-art user experience (see Chapter 2), it was made a constraint for the application to be able to dynamically guide the user through the flow of time, as can be seen in *Synthesia*, *iReal Pro* or *Vocal Pitch Monitor*, and also give feedback on the player's pitch (see Chapter 1.2). After noticing that all related works show a weakness in this area, it was defined as a goal to be able to help the player resolve progressions of chords to actual musical scales as well (see Chapter 1.1). This argumentation can be summed up to 3 main goals:

- Guide the user through the flow of time.
- Give the user feedback on their current pitch.
- Resolve chords to a full chord/scale representation.

The following sections in this chapter explain how the application was designed to meet these three goals and what choices were made to ultimately lead to the final design.

4.1. A Preview

Before diving deeper into the details of design choices for individual parts of the visualisation, the final design (see Figure 4.1) is anticipated so that the reader has an overview of how each of these parts fits into the "big picture" of our program. A screenshot of the design can be seen in Figure 4.1.

The three requirements defined in the beginning of Chapter 4 lead us to the conclusion that the application's visualisation needs to be two-dimensional, as scales and chords are related to pitch and vice-versa and can therefore be displayed in one dimension with pitch, while time needs to be displayed in a second dimension. Following

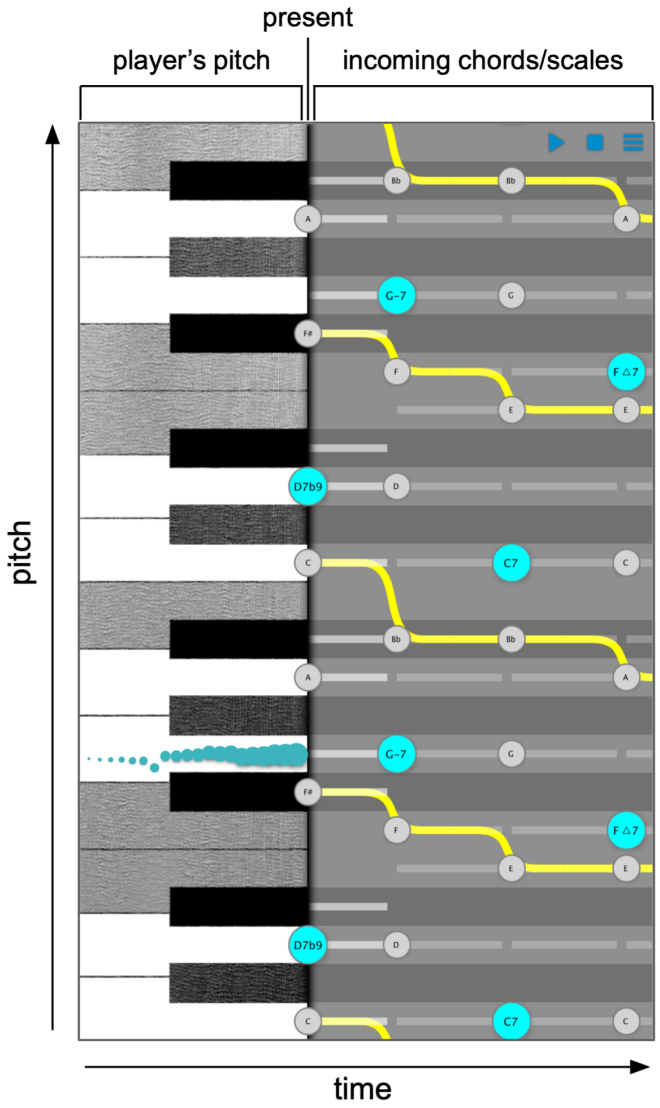

Figure 4.1.: A screenshot of the final product. New musical information appears on the screen's right side and then continuously moves to further left until it vanishes under/behind a fixated keyboard when it is due. This keyboard is used as a pitch reference. The player's current pitch – displayed as a group of turquoise circles that vertically follow a "head" – is displayed with its "head" directed to the right side of the keyboard so that it can easily be set in relation with present chord/scale information – in this case a *D7b9* chord. The background for incoming chords/scales is colored in the same pattern as the piano to provide the same pitch reference for the player's current pitch as well as for the pitch of incoming musical information.

the behaviour of typical staff notation (see Chapter 3) as well as lead sheet notation (see Chapter 1.1), the time is layed out on the x-axis with time increasing for increasing values, while the pitch – from low to high – is layed out on the y-axis. This is also conform to latin script, which is wide-spread across western contries and is read from left to right, and the common human behaviour of relating vertical height to pitch [21]. The way time and pitch are displayed is similar to how this is realized in *Synthesia*, except in our case, notes are not vertically "falling" but horizontally moving towards a keyboard. Also note that here, the musical information consisting of those notes is not meant to be imitated but as a suggestion to the user. To reference pitch, the typical pattern for a piano keyboard of partially alternating white and black keys is applied to the background – you can think of this as a "ruler". As mentioned above, this reference is used to relate both player pitch and harmonic constructs. Further – more illustrated – information about the general design can be retrieved from Figure 4.1.

4.2. A Suitable Keyboard Layout

As mentioned above, a piano structure – or so-called *piano roll* – has been chosen as the general primary pitch reference (see Chapter 4.1). This is yields the advantage that it might already be familiar to most users, but to a newcomer is not as complex to comprehend as the conventional musical notation system and is fit to avoid pitch ambiguity (see Chapter 1.1). For its design in the application-specific context, two requirements have been stated and must be taken into account.

A first requirement is that the keyboard has to fit a policy of treating all notes equally. This is mainly to be able to uniformly display intervals so that they keep the same proportion for all transpositions (see Chapter 3.1). We also remember (see Chapter 4.1) that we want to use the pattern of partially alternating black and white keys as a "ruler". This requirement can be reformulated to distributing all notes equally, which means that the height of a note is constant for all notes. Therefore one has to equally distribute the keys on the – from the perspective of a piano player – player-far side (on the player-near side, there are only white keys). Surprisingly, a close look at the proportions of real piano keyboards reveals that this criterion is not met by conventional keyboard layouts, as notes on the player-far side are not distributed equidistantly nor equally. A mathematical explanation would be that there are 12 keys in an octave, of which 7 have to be equally distributed on the player-near side, which is a desired criterion for a keyboard layout. If one wanted the 12 keys on the player-far side to be equally distributed as well, they would face the following problem: The white keys for the notes E and F are direct neighbours and therefore there should be a straight line between them. But – on the player-far side – the E is the *fifth* of *twelve* notes, while it is

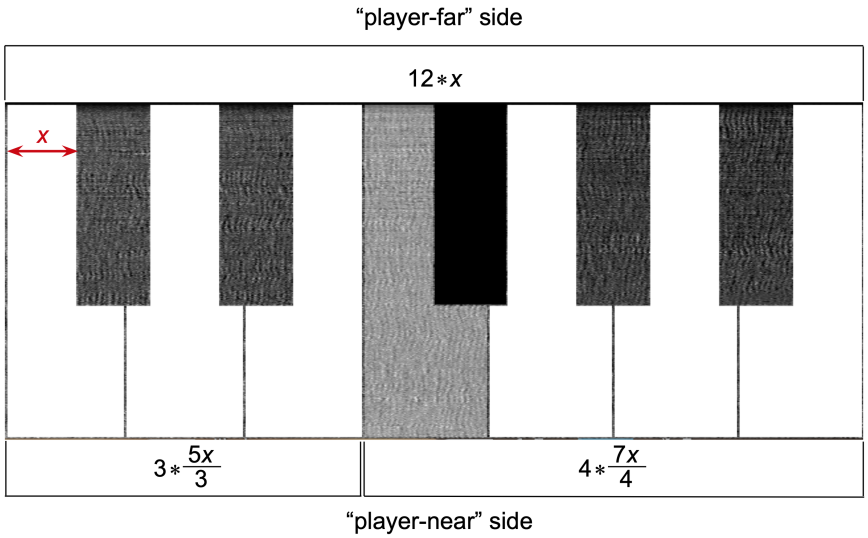

Figure 4.2.: A screenshot of the featured piano implementation: Note names *C*, *D*, *E*, *F#*, *G*, *A* and *B* are currently highlighted, therefore displayed fully opaque, while other notes remain transparent. An equal distribution of keys on the player-far side is achieved with a width of x for each of the 12 keys. To maintain a piano-like appearance, keys on the player-near side are split in two subgroups with differing widths.

the *third* of *seven* on the player-near side. These two quotients are close to each other – to be precise, they differ by $\frac{1}{84}$ – but they are not the same and therefore it would not be possible to have a straight line between the *E* and *F*. Our layout does not have the constraint of having equally distributed white keys on the player-near side, hence we can assign the same width to all - black and white - keys on the player-far side, and divide the white keys in two subsets – one from *C* to *E* and one from *F* to *B* – to at least achieve matching widths for those two subsets. This mandatorily results in unequal widths of white keys on the player-near side, but these unrealistic proportions are considered a good tradeoff to meet our requirement. You can see this method illustrated in Figure 4.2.

The second requirement is to be able to display – on our keyboard – which notes can be played in the current situation . The user is already visually provided with suggested notes (see the *D7b9* chord in Figure 4.1), however, it is desirable to further emphasize on the current musical suggestions. The keyboard section in this application's layout is a particularly suitable spot to do this, as this is also where the player's pitch is displayed, hence it is likely that the keyboard is visually focussed by the user. Displaying possible

note choices to the player could be solved by making a note on the keyboard appear pressed (this method is used e.g. in *Synthesia* [24]) , but here we rather want to present possible choices than already picked ones. This is why the piano keys are shown in transparent colors when not featured and are fully opaque while featured (see Figure 4.2). Using Opacity for this has the advantage of not being forced to introduce a new color or shape on the user interface, thus keeping the user interface simple.

4.3. Displaying the Player's Pitch

As mentioned in Chapter 1.2, pitch feedback is assumed to be useful for singers or many more instruments without discrete pitch values – strings, for instance. In our case, it is not only used for an end in itself, but to provide the user with an opportunity to compare their own melodic position with what notes the application suggests for them to play (see "present" in Figure 4.1). Note that, constricted through our pitch-detection algorithm, we are able to detect only note at a time, thus the detection is constricted to *monophonic* signals as produced by e.g. vocalists or wind instruments, while *polyphonic* sound sources like a piano cannot be detected unless the player chooses to play only one note at a time. Here we will present three different pitch visualisation approaches that have been implemented and evaluate them regarding their applicability for our purposes. One of them will display only the present pitch while the other two additionally display the pitch course over the recent past.

The former is inspired by a virtual keyboard representation as can be seen with *Synthesia*, where pressed keys on a controller keyboard are marked as pressed on the virtual represenation as well (see 2.3). But for the modest keyboard design we have chosen, such a *skeuomorphic* approach as featured in *Synthesia* would mean overshooting the mark in giving a the keyboard visualisation a 3D-like character. This is why we have chosen to outline the currently played key with color. To make pitch changes even more obvious to the user's eye, this outlining effect is enforced by a slight decrease in the size of the key played. We call this method *key* pitch visualisation (see 4.3).

The second presented approach is trying to display the current pitch together with a short pitch history of about half a second. Here, the present is on the very right side of the keyboard while past notes are displayed left of their predecessor. In theory, each single detected pitch in this short history is displayed as a circle on the keyboard. In practice, this is achieved by having those circles layed out on the keyboard with a fixed position on the x-axis and a fixed time distance to the present detected pitch for each individual circle. In the application's backend, a history of detected pitches together with a time signature indicating their detection time is refreshed whenever a new detected pitch is retrieved. Each circle's position on the y-axis is then determined

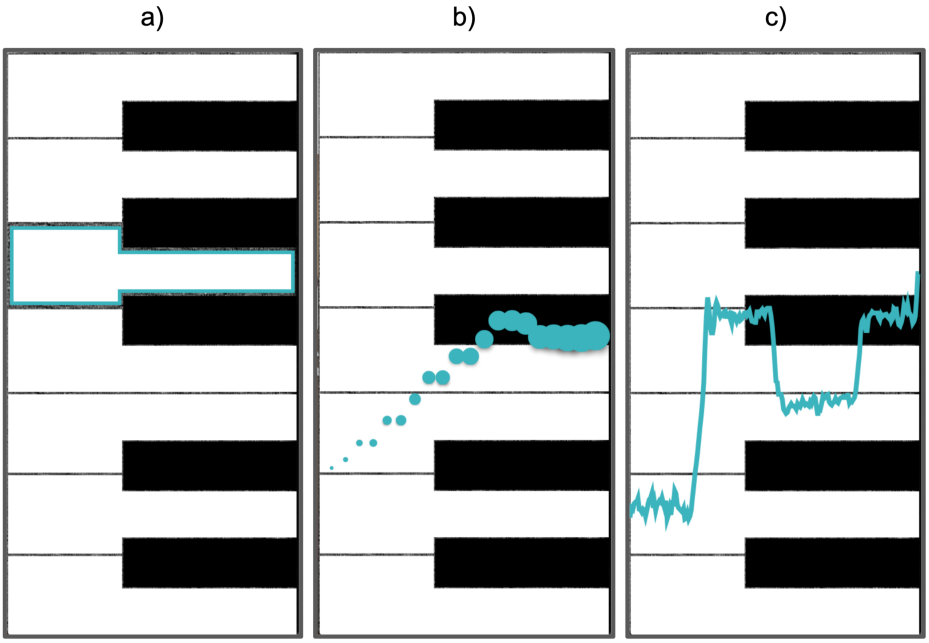

Figure 4.3.: a) The key pitch visualisation method. **b)** The bubble pitch visualisation method. **c)** The graph pitch visualisation method.

by assigning it to the pitch history entry that has a time distance to the present that is the closest to the circle's pre-assigned time distance. This can be understood as form of approximation. Because of how short this history's time is, it appears as a sort of "rat tail" to the current pitch, giving a visual sensation of "floating" or "hovering" over the keyboard. You can see this method, that we call the *bubble* pitch visualisation, in Figure 4.3.

The third approach is to draw a larger – multiple seconds – time frame of the abovementioned pitch history precisely as a graph. This approach is similar to what can be seen in *Vocal Pitch Monitor* (see Chapter 2.5). The time window is chosen such that this graph moves – horizontally to the left – in time just as fast as the harmonic information is moving towards the keyboard. We call this method *graph* pitch visualisation (see 4.3).

The *key* pitch visualisation can be further distinguished from the two others as it is only able to display pitch in a *discrete* way, while the other approaches are able to display *continuous* pitch values accross the keyboard. From this, the conclusion can be drawn that it is more suited for instruments with fixed pitches like piano or guitar,

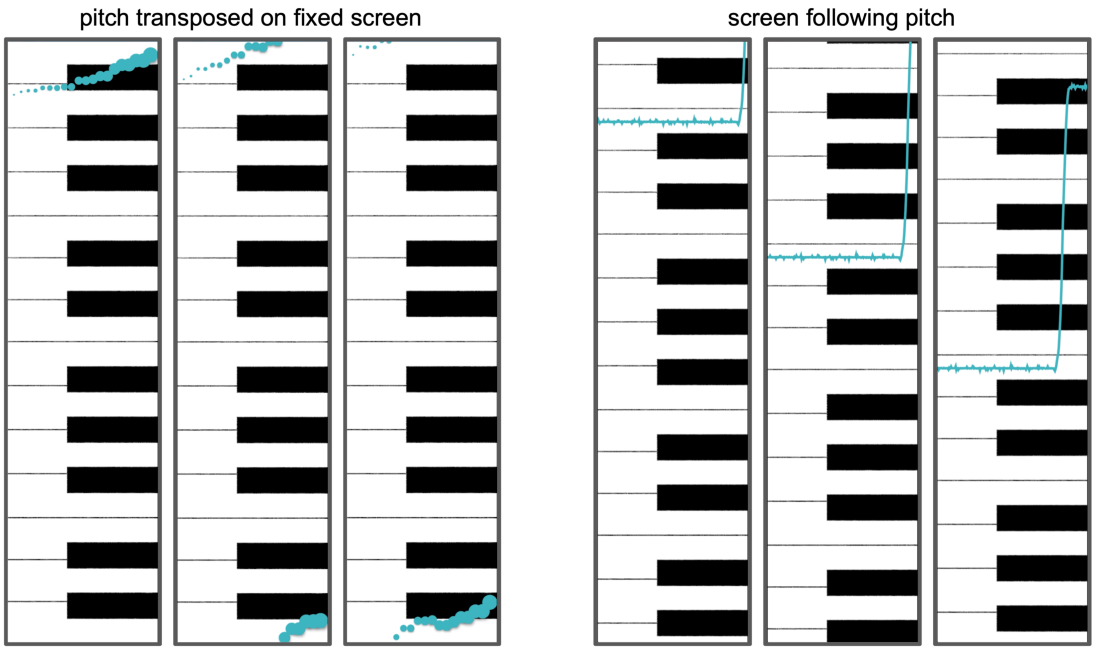

Figure 4.4.: Strategies to handle the event of the present pitch being outside of the screen's current pitch range. On the left side, the pitch is transposed to a lower octave. On the right side, the screen's pitch range is gradually adjusted to include the present pitch. Figure 4.5 shows, how this adjustment can be realized within a smooth motion.

while the *bubble* or *graph* pitch visualisation better at displaying the pitch behaviour of instruments which inherently produce continuous pitches like voice or trombone. Also, when a pitch lies in between two notes, the *key* pitch visualisation can often be seen flickering between those two notes. With the *graph* pitch visualisation, the desirability of such a long pitch history window is questionable as the user might – by peeking into the past – lose their focus on processing new incoming musical information. To conclude, this leaves us with a recommendation towards the *key* pitch visualisation method for instruments with a discrete pitch distribution and towards the *bubble* pitch visualisation method for continuous pitch distributions.

One problem all our proposed methods have in common is that it is always possible to produce pitches – either too low or too high – that are not in the screen's represented pitch range. Ignoring such events would mean that players lose their pitch indication. To solve this, we have pointed out two solutions. On an abstract level, it is either possible to modify the detected pitch in such a way that it can be represented within

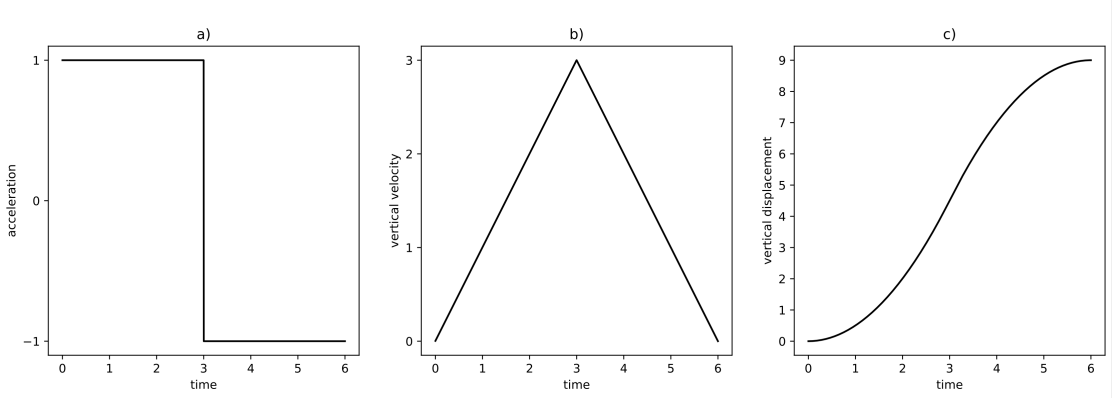

Figure 4.5.: Smoothly adjusting the screen's pitch range offset using an acceleration constant. At $time = 0$ an out-of-bounds pitch is noticed, which is 9 pitch units too high for the current screen pitch range. **a)** The screen's pitch range offset begins to accelerate, **b)** velocity starts to increase linearly. **c)** This leads to a quadratic shift in the screen's pitch displacement. At $time = 3$, by integration over the future velocity – thus for $time \in [3, 6]$ – it can be anticipated that the desired pitch will be caught up soon and a breaking process is initiated by reversing acceleration. The desired pitch distance is covered at $time = 6$.

our screen's pitch range or – the other way round – to modify the screen's pitch range offset such that the range includes the out-of-bounds pitch.

For the first, this can be done by transposing a pitch by one or several octaves up or down in order for it to be in the screen's pitch range. For our use case, this yields no loss in musical information – for example, a *C#*, transposed by one or several octaves, always stays a *C#*. Such a "wrap-around" can be seen in Figure 4.4.

A solution where the background is adjusted rather than the pitch can be seen in *Vocal Pitch Monitor*. We have implemented a similar functionality that can be understood as the background "chasing" after the pitch (see Figure 4.4). To make the screen move in a smooth way, its velocity has to increase when the pitch goes goes out of bounds and to decrease *before* the pitch is caught up. It is possible to do this utilizing an acceleration constant (see Figure 4.5).

While a fixated pitch area has the advantage of visual clarity in every situation, a pitch "wrap-around" will behave adverse to what the user expects in the event of the pitch "leaving" the screen. On the other hand, a moving background can take away from visual clarity – it is harder to contextualize the current pitch with incoming chord/scale information, when the screen is moving – especially if there is background noise involved in the player's environment. This noise can lead to frequent events

of misdetected pitch, which can then lead to a background screen that is constantly moving.

4.4. Synchronized Flow of Sound and Image media

Given a backing track for a song, the song's visualized harmonic material must be moved accross the screen in a synchronized way, as this goal was stated in the beginning of Chapter 4. For reference, this movement concerns the "incoming chords/scales" section in Figure 4.1. Remember that, as described in Chapter 4.1, the basic idea of how harmony is displayed over time is inspired by *Synthesia*'s idea of "falling notes" [24], but here we apply horizontal rather than vertical movement.

In order to synchronize the visual representation of a song to its backing track in real time, one has to know how both flow through time. While the progression in an audio file is measurable in *samples*, the visual representation's respective progression will in our case be measurable as a value on the x-axis (horizontal dimension). Given the current sample position of an audio file, it is easy to convert it into a time value if the *samplerate* – measured in samples per second – of the audio is known. In order to correlate the visual representation of the chord/scale progression with time values, we need to take a closer look at the song data it is representing. As can be seen in Figure 4.7, each chord is stored with a parameter indicating its length in musical *bars* and that there are global parameters indicating *beats per minute* and *beats per bar*. From these indications, given a sample position in our audio track, we can now measure how far our visual representation must have progressed to this point in time.

> **Example:** To demonstrate this, we use values that correspond to Figure 4.7. Assuming our backing track plays at a standard samplerate of $44100Hz$ and is currently playing the sample at position 176400, this tells us, that we are $\frac{176400}{44100Hz} = 4$ seconds or $\frac{4}{60} = 0.0\overline{6}$ minutes into the track. To find out how many musical *beats* this corresponds to, we compare that we are given 90 beats per minute but only $0.0\overline{6}$ minutes of playing time, therefore indicating that – up to this point – exactly $90 \cdot 0.0\overline{6} = 6$ beats have passed. With 4 beats per bar this tells us that we are $\frac{6}{4} = 1.5$ bars into the song. The chord of *Fmaj7* (length of one bar) has already expired and we are halfway into *D7b9* (also one bar). In Figure 4.7, the application displays a time window of two bars in its x-axis range reserved for song representation. We can now find out that at sample position 176400, the visual representation of our song has already shifted to the left by $\frac{1.5}{2} = 0.75$ portions of this area.

Additionally to this linear relation of time and visual displacement, we have implemented a second approach to moving chord/scale information accross the screen, that

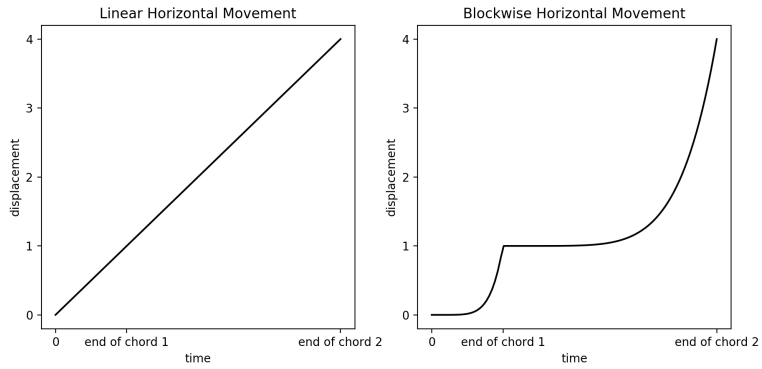

Figure 4.6.: Linear and "blockwise" representation of harmonic flow over time. On the left, one can see that the horizontal displacement of chords/scales is always "true" to the progress in time. The right graph shows a movement pattern where – for the most part of a chord's duration – chords/scales barely move over time. However, just before a chord/scale ceases and is replaced with new harmonic content, this lack of movement is compensated through accelerating to a polynomial displacement curve.

aims towards a movement that is visually perceived as more calm. The idea is that as long as the current chord is being played, its representation is "pinned" right next to the keyboard, where our pitch reference lies. However, to prevent our movement from degenerating to what could be described as a slideshow-like behaviour, a polynomial curve was used to approximate this idea (see Figure 4.6). This way, it looks like our visual representation is being processed block by block rather than in a continuous fashion, but the movement still clearly indicates the event of the current chord being replaced by a new one.

4.5. Displaying Chords and Corresponding Scales

This section describes how a song's chord- together with scale-information is visually represented. As stated in Chapter 2.6, our chord representation not only has the aspiration to break down chord symbols to chord notes, but also to guide the user in their scale choices for respective chords (see Chapter 3). In Figure 4.7, one can see how this data is mapped to the display in the final version of our application. We will now present basic design choices regarding this representation before introducing and discussing two implemented approaches to visually present chord/scale information. Note that the pitch context our chord/scale representation is built upon is already presented in Chapter 4.2.

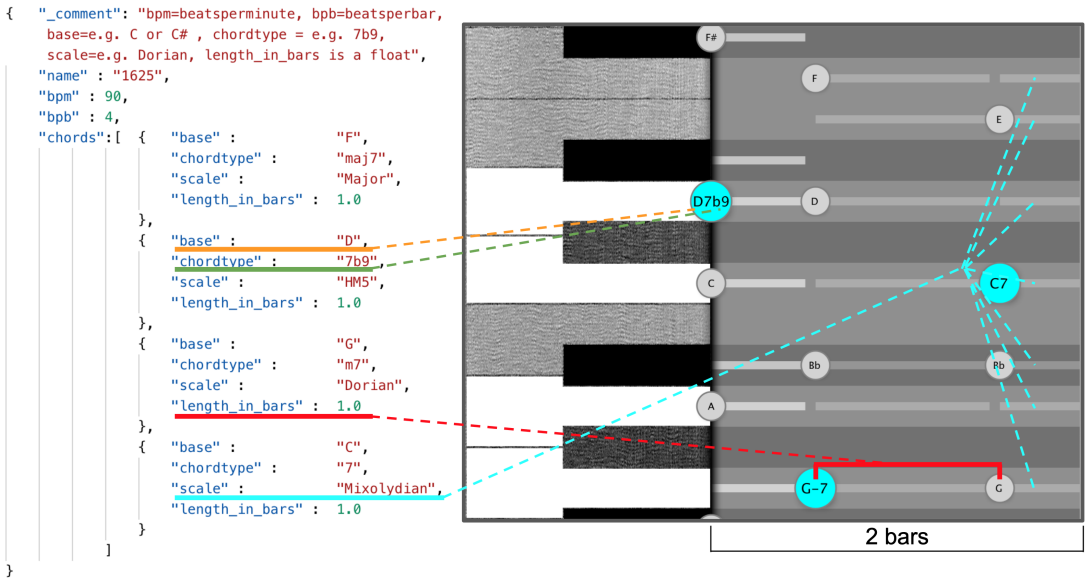

Figure 4.7.: On the left, one can see a song stored as data in *JSON* format. Next to global parameters like the song's name, *bpm* (beats per minute) and *bpb* (beats per bar), the most relevant information for an improviser is the chord sequence. Each chord will be played a certain amount of time (length), has a certain type, a base note it is built upon and – a main feature of this application in contrast to related work – a scale that can be utilized to play over this chord. On the right, one can see how each of these chord parameters is displayed in the application.

As described in Chapter 2, Jamey Aebersold's book series and the application *iReal Pro* both approach jazz improvisation utilizing leadsheets. In this respect, they also have in common that both offer features to break down chords and scales into musical notation. In Chapters 2.6 and 1.1 we have explained why we want to take a different path than standard music notation. Despite the use of musical notation it is observable in Figure 2.1 and Figure 2.2 that the scale-representation features used in *iReal Pro* and Jamey Aebersold's work both distinguish chord notes from notes that are included in the scale as well as the chord. This makes sense because, as already mentioned in Chapter 3, improvising on an entire scale yields more harmonic tension than improvising only on chord tones. Sikora also mentions the importance of conscious decision-making between chord notes and *"upper structures"* (notes that are included in the scale but *not* in the chord) [22, p. 90]. Furthermore, as we have seen in Chapter 3, the chord notes for a chord symbol are unambiguous, while other scale notes or "upper structures" can

be adjusted to the player's liking – of course always considering musical context. For these reasons we have chosen to separate chord- and scale-representation as well.

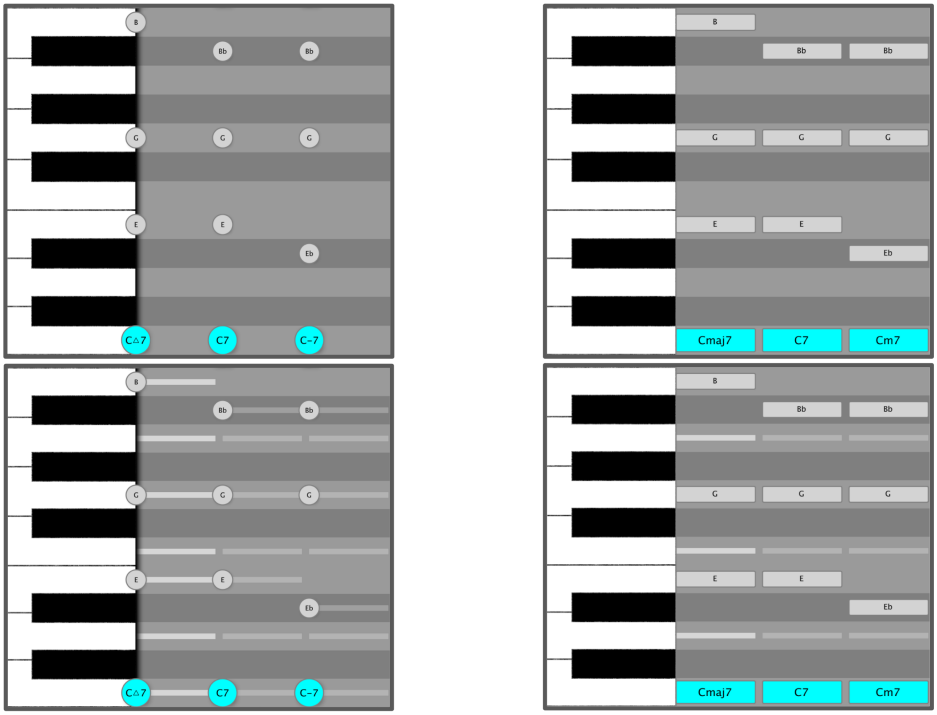

Figure 4.8.: The dot visualisation method (left) next to the bar visualisation method (right). Both display different chord types with the same vertical proportions. The bar visualisation method displays chords in a broader horizontal range and therefore yields the possibility to use less compact chord symbol representations. One can see that both approaches behave differently when scale notes are displayed (bottom section). While, with dot visualisation, chord notes have the same light-grey horizontal bars as other scale notes, bar visualisation displays a clearer visual distinction over the entire horizontal span of a chord.

In order to represent chords, a first method that was implemented is to display chords as bars. Here, we call this approach the *bar* chord visualisation method. Obviously, chord notes are vertically aligned to the keyboard grid, resulting in the chord type's inner proportions becoming apparent to the user's eye (see Figure 4.8). All such bars are displayed with the respective note names written on them to further clarify their pitch values to the user. Root notes are coloured, vertically thickened and display the entirety of a chord symbol. Note that such a chord symbol is all a user would be able

to see on a lead sheet. The presented bars look similar to *Synthesia*'s approach on displaying *MIDI* notes (see Figure 2.3). Also like in *Synthesia*, the bars currently played gradually disappear behind the keyboard.

As Figure 4.9 illustrates, this mechanism also introduces the problem of chord symbol- and note text disappearing behind the keyboard together with the bars they are written to. For this reason, we have implemented a second method to display chords, the

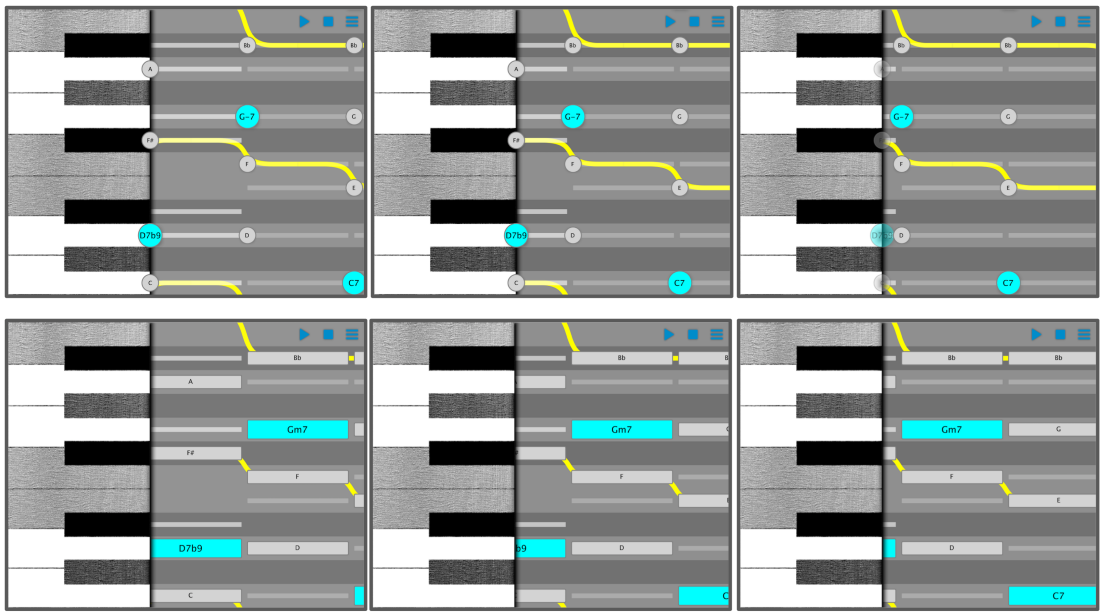

Figure 4.9.: In the bottom section of this figure, one can see the chord symbols gradually disappear for the current chord when using the bar visualisation method. As can be seen in the top section, the dot visualisation method solves this problem by "sticking" its dots for the current chord to the keyboard's far end and letting them disappear through a transparency fadeout. The dots for the current chord disappear shortly before the chord is replaced by another chord as the current chord.

dot visualisation method. It has the same properties as the bar visualisation method regarding vertical layout and chord note or chord symbol display, but chord tones are drawn as round circles instead of as bars. Their horizontal appearance is therefore more narrow, thus chord symbols have to be written in a smaller format (see 4.8). This can be come by by using an abbreviated chord symbol layout that is commonly used in lead sheets and can be seen in Jamey Aebersold's *Nomenclature* [11]. The dot visualisation solves the problem of disappearing chord symbols with dots that are not disappearing behind the keyboard, but rather sticking to the keyboard and fading out

into full transparency right before their respective chord is ending, as can be observed in Figure 4.9. By using blockwise horizontal movement (see Chapter 4.4), the problem can be solved for the bar visualsisation method as well. In this way, by not horizontally moving for a large portion of a current chord's time span, the chord is prevented from disappearing under the keyboard up until the chords are exchanged through quick movement.

The main difference between bar and chord visualisation can be observed considering the topic of visualizing scales. Compliant to the bar visualisation method, we have chosen to represent scale information as bars. This choice was made because it is of interest for the player to be able to visually perceive harmonic changes, which are reflected by changes in the respective scales (see Chapter 3). In order to achieve this, the player needs to be able to compare the current scale set to the set(s) of incoming chords at any point in time. The scale bars are visualized such that they horizontally represent their respective chord's length in time and therefore such a comparison is possible at any time of the chord's timespan. This can be observed in Figure 4.9, where, even right before a new chord/scale set is introduced, a player can compare the current set to its subsequent set. They might, e.g. , come to the conclusion that, in Figure 4.9, the *Eb* the *D7b9*'s chord representation suggests is not going to be a suggested for the subsequent *Gm7/G-7* chord.

To be able to distinguish between chord- and "upper structure" notes, the scale notes are displayed more transparent and vertically slimmer than chord notes. The dot visualisation method's dots can be seen more as a "point in time" than bars in the bar visualisation method, which manages to display the entirety of a chord's timespan. The presented scale visualisation method provides a way to cope with this flaw as it "extends" the dot visualisation's displayed timespan horizontally, but still, the hierarchical difference between chord- and "upper structure" notes can be perceived more easily throughout the timespan of a chord using the bar visualisation method (see Figure 4.8).

As mentioned in in Chapter 3.4, guide tone lines, while sounding engaging to the listener, are a relatively simple method of outlining harmonic material. Therefore, guide tone lines are a standard practice routine for jazz musicians, especially when they are trying to learn and understand the harmonic patterns in a song that is new to them. For this reason, we implemented a feature to present guide tone lines next to the chord/scale representation. Their look is inspired by the idea of a path *guiding* a user's way through harmony. According to this, guide tone lines are visualized as paths that connect the *sevenths* and *thirds* in a sequence with each other. This could be best described as "connecting the dots". They are chosen to be vertically just as slim as the grey bars indicating scale notes, but are highlighted in yellow colour so that their visualisation can live up to their suggestive purpose. As Figure 4.8 illustrates, the

featured guide tone line implementation has more visual presence in combination with the dot visualisation than when combined with bars.

5. Future Work

In this chapter, we present what must be done to make the application presentable to a larger audience. For example, the application can not yet be used on other platforms than *MacOS* and must be adjusted for the use on smartphones, where especially the smaller screen size must be considered in order to provide a suitable user interface. However, these are minor adjustments as opposed to what has to be done to provide an application that can catch up with the technological state of the art.

As anticipated in Chapter 1.2, current state-of-the-art software solutions for musical guidance provide some features we have not implemented yet. These features include the possibility to practice only a selected section of a song, to transpose songs or to practice a song in a slower/faster tempo. While selecting a specific song section is considered as relatively easy to implement – one has to narrow the set of visualised chords and the sample range in the audio playback accordingly – changes in transposition and tempo force a song's given audio playback to be mutated. This could for example be done utilizing pitch- and audio stretching methods based on the *Waveform Similarity Overlap-Add* (WSOLA) time-scale modification algorithm, which is well-documented and available as open-source software [5]. However, *iReal Pro* provides these features in a more sophisticated way. As mentioned in Chapter 2.2, backing tracks are dynamically generated by a virtual band. Parameters like musical style or instrumentation of the band can be adjusted to the users liking. This also means that the applications memory size regarding audio data is able to stay constant when new song information is loaded from a database, as audio data – except audio samples used by the virtual instruments – is not required to be able to create backing tracks to any song. These advantages outweigh the fact that implementing such an environment would supposedly be more time-consuming than using pre-implemented audio manipulation algorithms.

This brings us to the topic that – like in *iReal Pro* – song data would have to be loaded from a database or other online resource to the user's device. Such a mechanism also opens up the possiblity for users to create and share song data via the database, which could over time lead – as it is the case with *iReal Pro* – to a large and profound corpus of song data available to all users of our application. At this point the question can be arised of why we even need a new set of song data as this is already provided by corpuses like *iReal Pro*'s database or the *iRb* [3] corpus and others. The answer is that these resources offer song data that is annotated with lead sheet notation in

mind. Songs are basically stored as sequences of chord symbols subdivided into a song's sections. The *iRb* corpus for example additionally provides computer-generated functional harmony annotation, but such resources do not provide *scale* annotation together with these chords. For our application domain, scale annotations as illustrated in Figure 4.7 are mandatory in order provide an improviser with scale information. Therefore we have to find a way to accumulate scale-annotated chord data. We have pointed out in Chapter 3 that finding scales to given chords is a task that – besides leading to ambiguous results – requires a certain degree of expertise in musical analysis and listening experience. When a chord/scale analysis like in Chapter 3.3 leads to ambiguous results, it is ultimately a matter of choice for the musician to opt for the use of one specific scale. Such choices are often dependent on the musical style – there are several subgeneres to jazz music such as *Funk* or *Avantgarde* – at hand. Therefore they are rather inept for the purpose of being formalized into a computational approach. To make sure to provide valid chord/scale information to a user, a user-driven approach like it is the case for *iReal Pro* seems like the best solution.

5.1. Adapting Natural Language Processing to Scale Estimation

However, at a hypothetical point in the future when we have successfully built up a database with a wide range of the most common jazz standards stored in it, users without expert knowledge about scale analysis might still run into difficulties. Consider the scenario of an inexperienced musician who wants to play and improvise over a lead sheet that is *not* stored in the database. From the lead sheet, they would have chord information at hand, but would not be able to translate this into scale information without expert help. For this reason it would be desirable to implement a method to be able to automatically provide the user with scale annotation to the given sequence of chords.

Given our hypothetical dataset, it would be possible to refer to its pool of annotated chord/scale data in order to compute a "best guess" for the unknown chord sequence. To implement such a method, we want to lend an approach from the field of *natural language processing* – from the field of *Part-of-Speech tagging* to be precise [19, p. 341]. In short, part-of-speech tagging is the task of assigning words in a sentence to a certain word category based on its context. For example, the term "will" is assigned to different word categories for the sentences "I will survive" and "This is my will". It serves as an auxiliary verb in the former sentence while being a noun in the second. In order to computationally distinguish between such categories, the constellation/context a term appears in is statistically compared to a pre-annotated set of data. Therefore, all permutations of word category sequences the word sequence of a sentence could

possibly be assigned to are evaluated regarding their statistical probability. Now, since each such permutation is a sequence of word categories, the statistical value given to the permutation is then multiplied by the probability for each word category in its sequence to actually produce the given word. The permutation that yields the highest probability value for this maximum-likelihood estimation will then determine what categories the words in a sentence are assigned to.

Example: To analyse the sentence "I will survive" according the above-mentioned method, we first need to determine what categories each of the words in the sentence could be assigned to. For simplicity, aussme that "I" can only be a noun and "survive" can only be a verb, while "will" could be a noun or an auxiliary verb. This leads us to two possible tag permutations: (noun, aux-verb, verb) and (noun, noun, verb). The probabilities of those two sequences of word categories to appear in the annotated dataset can be approximated using a Markov chain with *bi-grams* [19, p. 345]. The probability of a certain word category appearing in a text supposing its predecessing neighbour is of a certain other category is used as a transition probability. In our example, this can be translated to the following conditional probabilities:

noun	aux-verb	verb
$P(N\|start)$	$P(AV\|N)$	$P(V\|AV)$
0.7	0.2	0.4

noun	noun	verb
$P(N\|start)$	$P(N\|N)$	$P(V\|N)$
0.7	0.3	0.6

Note that as there is no left ancestor to the first word, a word category's probability to be representative for the first word of a sentence is used here. Also note that for demonstrational purposes, values are used that are made up and unrealistically large. For the two possible interpretations, the sequence probability is:

$$P((noun, aux\text{-}verb, verb)) = 0.7 \cdot 0.2 \cdot 0.4 = 0.056$$
$$P((noun, noun, verb)) = 0.7 \cdot 0.3 \cdot 0.6 = 0.126$$

Now, the given evaluation method proceeds to consider the probabilities for each suggested category and position to have produced the respective word instance.

$P(\text{"I"}\|N)$	$P(\text{"will"}\|AV)$	$P(\text{"will"}\|N)$	$P(\text{"survive"}\|V)$
0.6	0.6	0.2	0.2

Multiplying the previously calculated probabilites for category sequences by these category-word relations leads to the following end results:

$$P((noun, aux\text{-}verb, verb)|\text{"I will survive"}) = 0.056 \cdot 0.6 \cdot 0.6 \cdot 0.2 \sim 0.004$$
$$P((noun, noun, verb)|\text{"I will survive"}) = 0.126 \cdot 0.6 \cdot 0.2 \cdot 0.2 \sim 0.003$$

The values used here lead to a correct result, as (noun, aux-verb, verb) is more likely to be the correct tagging according to the given dataset.

The word-to-category approach used in part-of-speech tagging can be adapted to music as a chordtype-to-scale approach. From our given chord set we use chord *types* rather than chords, because – in contrast to entire chords – they are not dependent on the base note they are transposed to. The chord types can then be assigned to a set of scales as demonstrated in Chapter 3.3. In our approach, scales are used like categories in part-of-speech tagging. The only difference to the procedure from the above example is, that we now have to find a way to take into consideration the intervallic proportion the base notes of the given chords have to each other. This is done by including them into the bi-gram probability from the first evaluation step. Recall the chord progression used as an example in Chapter 3. The transition function from *C7* to $F\triangle$ would then – taking into account the falling fifths movement – defined as $P(Major|Mixolydian \wedge \downarrow 5)$ and for the second evaluation step, $P(7|Mixolydian)$ and $P(\triangle|Major)$ would have to be retrieved from the hypothetically exhaustive dataset.

5.2. Real Time Challenges

Another future challenge for the application developement is to meet real time requirements that are introduced through the use of pitch tracking. Therefore, the timespan between a pitch generated by the user and it being displayed on the screen must be minimized in order to not disturb the user experience. This latency depends on:

- The time it takes for a computer/tablet/phone or other device to record audio from the microphone into the system such that it can be processed
- The time it takes to calculate played pitch for a pitch tracking method
- The time it takes until the calculated pitch is rendered to the device's screen

Claypool and Finkel's study *The effects of latency on player performance in cloud-based games* [4] has shown that user performance declines significantly when the overall visual latency in a computer game exceeds the mark of 100 milliseconds. With that in mind, we set out a goal to stay below this mark. However, our analysis shows that at this point in time, such a performance criterion cannot be guaranteed. State-of-the-art mobile devices are mostly below the 40 milliseconds mark – the best-performing models even reach about 10 ms – regarding their audio roundtrip time (the timespan as described above), but this value differs from device to device and can occasionally expand to more than 50 milliseconds [23], especially on Android devices [7]. This is a parameter that is not under our influence and we therefore make an assumption that audio roundtrip on

user devices is 50 milliseconds at most. After sound is recorded, a real time pitch tracker will compute its pitch. However all state-of-the-art pitch tracking algorithms handle audio data in chunks which represent a certain time window of buffered audio [14]. This can be explained as it is desirable to at least have one period of a signal's oscillation recorded in order to correctly calculate its pitch. Commonly used time window sizes for this purpose are of about 25 ms or – for lower frequency signals like male voice – of about 50 ms in time [14]. Like roundtrip latency, the pitchtracking latency cannot be optimized further. Assuming a device with high roundtrip latency and a user with male voice, we reach a summed up worst-case latency of about 100 ms before the retrieved pitch value is even handed over to the rendering thread. As to this point we are already about to overshoot the mark of our performance goal, we have to try and minimize the rendering time window, as in a worst case, a value twice as high as the rendering time would as well have to be added to the sum (see Figure 5.1). Measurements on our current application have shown that, with the current implementation, the period from the completion of one rendering call to the completion of the next call will be between 40 ms and 120 ms on a Macbook Pro from Mid 2014. This has to be tested also on other devices, but we assume that it might be possible to cut rendering latency in our application by utilizing the rendering environments provided by *OpenGL* [12] or *Metal* [2].

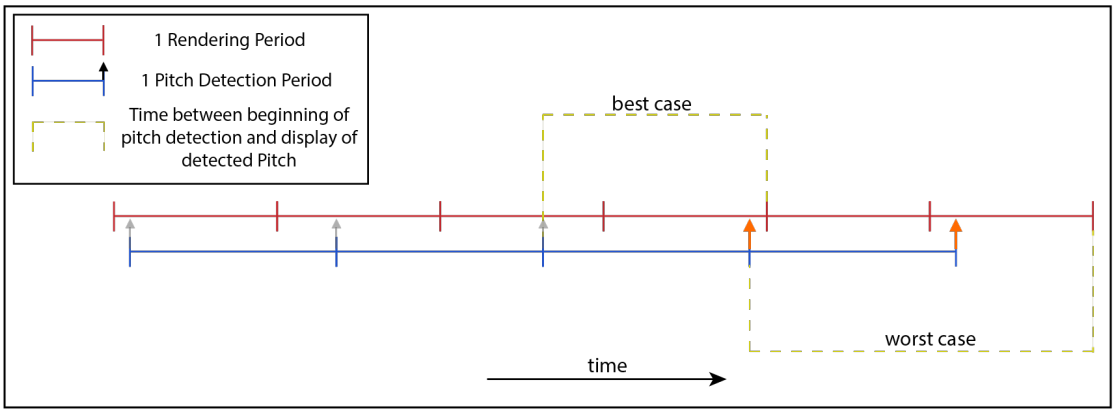

Figure 5.1.: An illustration of best- and worst-case scenarios regarding event latency between periodic concurrent processes. In this example, the rendering process is always displaying the latest calculated pitch value given to it by the pitch detection process. In the best case scenario, the pitch calculation is ready right before a new rendering call. The new rendering call will then display the given pitch when it is finished. The latency added after the pitch is calculated would thus correspond to one rendering period. In the worst-case scenario, the pitch calculation finishes right after a finished rendering call and is not fetched for a large proportion of a rendering period. After it is fetched, the delay of another rendering period is added to the latency.

6. User Study

To evaluate the result, particularly the design possibilities in our implemented application, we conducted a short user study with 7 participants. The participants were chosen such that their musical experience ranges from an amateur level (skills in playing an instrument and reading sheet music, but no deeper knowledge of jazz theory) to professional musicians with a college degree in the field of music. In the following chapter, the evaluation process and design of the questionnaire will be presented followed by an evulation of the study's results.

The study was designed with two goals in mind. On the one hand, to evaluate if a user's interaction with the application can improve their musical training and knowledge. On the other hand, to investigate which visualisation choices lead to a pleasant and – for music learning – convenient user experience and what could be further be optimized according to the users. The obvious solution to collect these informations was to make the test persons familiar with the software at first. We then let the user interact with the application on their own before giving them a questionnaire to fill out.

6.1. The Questionnaire

This questionnaire can be found in Appendix A.2. It consists of personal questions, questions about the presented visualisation approaches (see Chapter 4) and general questions about the application.

The personal questions were asked mainly in order to determine the user's previous knowledge in music and jazz improvisation. We asked for a test person's age, formal musical qualification and experience with improvised music. The next part of the questionnaire aims towards comparing different visualisation approaches for the topics of pitch tracking and -visualisation, keyboard visualisation and chord visualisation/movement. For each of these fields (e.g. pitch visualisation), the implemented approaches (e.g. key/bubble/graph visualisation) were evaluated against each other, utilizing the same set of questions for each. As it is a commonly suggested method for user experience studies, these questions were constructed such that they depict pairs of attributes and their respective antonyms and a user can decide between them on a scale from one (antonym) to five (attribute) [15]. After this comparison, users were

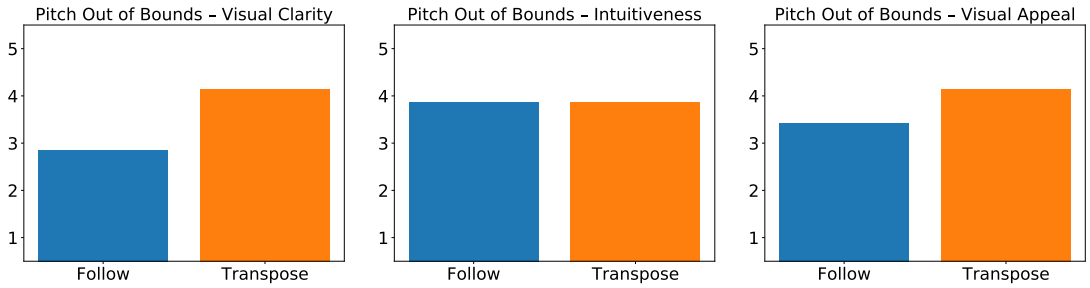

Figure 6.1.: The two strategies for handling an pitch out-of-bounds event, as presented in Chapter 4.3, rated by users regarding achieved visual clarity, intuitiveness and visual appeal.

optionally asked questions about possible improvements and flaws of the application's approach in the given field. The "general questions" section aims to gather the user's opinion on the application's value in musical education and leaves room for further suggestions. The exact answers to the questionnaire can be retrieved from Appendix A.1.

6.2. User Opinions on the Implemented Visualisation Methods

In this section, the given results to the questionnaire are evaluated. The first set of questions shows that all participants play or have played an instrument and most but not all of them have practice in music improvisation to a certain extent. This is according to what we have defined as our target user group.

The first field of visualisation evaluated in the questionnaire was the way our application displays pitch (see Chapter 4). As depicted in Figure 6.1, users rated the positive learning effect achieved by our real time pitch visualisation as above the indifference mark of three. While all visualisation approaches were ranked as intuitive rather than unintuitive, the key- and bubble visualisation method were ranked to be more intuitive than the graph visualisation method. The evaluation on visual appeal reveals a similar tendency, but users were indifferent regarding the graph visualisation's aesthetical appearance. It was mentioned that – in general – the pitch tracking appears to be "shaky", thus the pitch detection's tendency towards minor pitch changes could be smoothed out. In the event of pitch being too high or low to be represented on the keyboard, users preferred the method of transposing pitch rather than letting the screen follow. This was mainly for reasons of visual clarity and appeal. A closer look at Figure 6.1 reveals that users reacted slightly confused when the screen was chasing

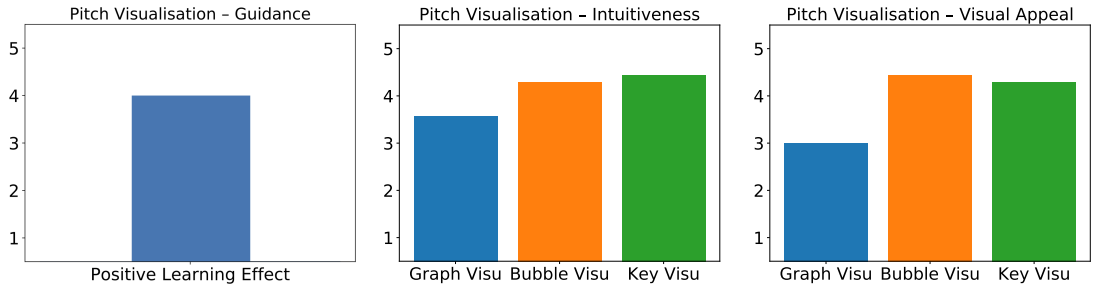

Figure 6.2.: On the left, one can see how users rated the effect of pitch visualisation on their musical intuition. The two diagrams in/on the middle/right show the three pitch visualiation methods discussed in Chapter 4.3, rated by users regarding their intuitiveness and visual appeal.

after the current pitch.

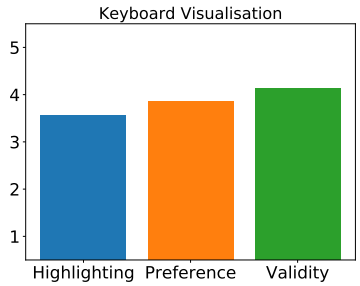

Figure 6.3.: The implemented keyboard visualisation and its use as a global pitch reference as presented in Chapter 4.2. It was rated by users regarding its validity as an approach for the given application domain. Furthermore, the users were asked to rate the helpfulness of note highlighting on the keyboard and if they prefer this pitch reference over staff visualisation.

The keyboard visualisation was subject to questions regarding its validity as a pitch reference for musical constructs, which could be approved 6.3. When asking users if they would prefer traditional staff notation over the presented piano roll notation, most users had a strong preference towards the piano roll, while especially trained musicians had a preference for staff notation. The highlighting mechanism of the keyboard as illustrated in Figure 4.2 was perceived rather indifferently in its ability to highlight the current scale through transparency. However, one participant mentioned that this highlighting could be improved to be more helpful to the user by applying coloring schemes instead of transparency.

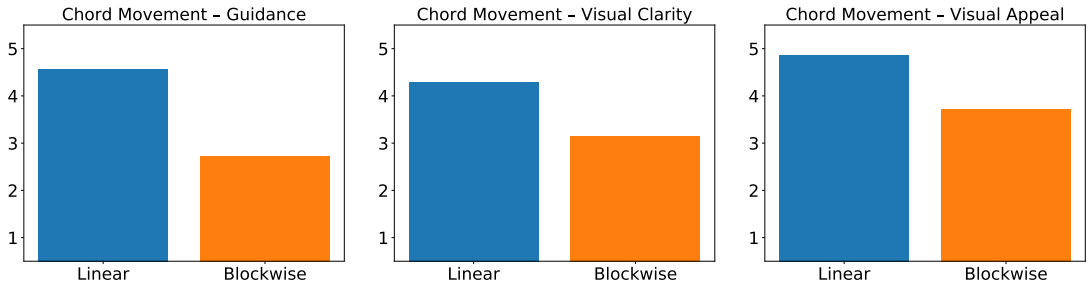

Figure 6.4.: The two approaches for displaying the flow in time horizontally presented in Chapter 4.4, rated by users regarding their effectiveness in guiding the user through the musical progression over time as well as visual clarity and appeal.

In Chapter 4.4, we have presented two approaches to translate time flow into horizontal movement on the screen. Contrary to our assumptions, linear movement was found to provide more visual clarity while some users found the block movement to be confusing rather than calm (see Figure 6.4). The straight-forward linear approach was also perceived better regarding its visual appeal and its ability to guide the user through a flow of time.

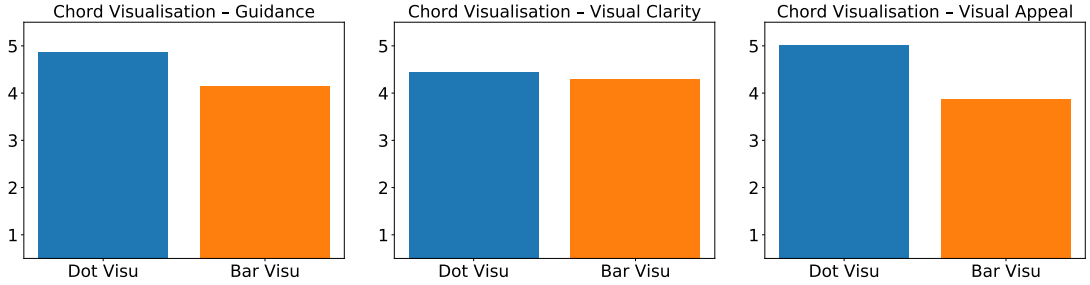

Figure 6.5.: The two approaches for displaying chords as presented in Chapter 4.5, rated by user regarding their suitability for melodic guidance as well as their visual clarity and appeal.

Both methods in the field of displaying chords were very well perceived by the users with better ratings for the dot visualisation on its ability to guide through the melodic flow and its appearance (see Figure 6.5). Not surprisingly to the above observations, all users except one ranked a combination of dot visualisation together with linear movement as their preferred configuration for a scale representation together with a horizontal movement approach. The study showed that users can have difficulties with

interpreting the vertical proportions of chords as chord types (see Figure 6.6), even when looking closely. One user suggested to solve this problem by color-coding the various chord types. The visualisation of guide tone lines was perceived as particularly appealing and clear to the user.

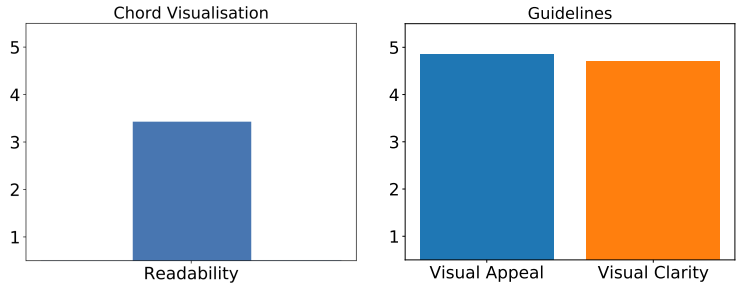

Figure 6.6.: On the left, one can see how users rated readability of the chord visualisation with an eye towards its ability to vertically display the proportions of certain chord types. The diagram on the right shows how the visualisation of guidelines (see Chapter 4.5) was rated regarding its visual appeal and clarity.

6.3. General Questions

As mentioned before, the general questions section was meant to evaluate the application's value for musical education. The users were positive that learning music improvisation is a difficult task without the support of a teacher/mentor and that the application provides new opportunities for self-teaching musicians without profound theoretical knowledge. One user pointed out that the application yields the opportunity to interactively experience musical concepts that would otherwise be hard to formally declare. The users also answered that they would suggest the implemented software to peers, if those had a certain aptitude to music or software applications in general.

Regarding improvements to the current implementation, users suggested adding a guitar visualisation or staff notation as additions to the current keyboard visualisation. Several users suggested that a virtual piano (playable on touch-screen devices) or connected piano controller could enhance the user experience and make the application less reliant on the use of real instruments. This approach could open up the opportunity for users, who do not know how to play an instrument, to experience jazz improvisation.

7. Conclusion

The work in this thesis aimed towards providing an educational software solution for jazz improvisation that introduces an alternative notational approach to musical lead sheet notation. We found that lead sheet notation is a valid format for denoting jazz standards, but that it also hides information that could be useful for inexperienced musicians trying to learn improvisation. We then evaluated methods to help musicians discover this hidden information – namely musical scales – and found that these rely on sheet music notation (see Chapter 2).

As sheet music reading requires musical training and knowledge and as sheet music can behave rather unintuitively in the way it displays musical proportions (see Chapter 1.1), we decided to take a different path in our software application. For our notation, we defined a piano roll as the main pitch reference and then implemented a chord/scale representation respectively. The implementation from Chapter 4 presents a range of possible designs for our notational method and additionally shows, how a user experience in guided jazz improvisation is provided. The main challenges here were to display a player's pitch in real time and to guide the user through the flow of time.

The successfully implemented approach was then evaluated in a user study (see Chapter 6). For a wide range of aspects in the visualisation and the notational system, the implementation was well perceived by the users and the users approved our assumption that there is a need for approaches in jazz education that do not presuppose formal musical training. From the user study, we also learnt that the presented notational approach could still be further optimized in such a way that chord types are easier to perceive. Furthermore, users suggested that – by including a playable touchscreen keyboard – our solution could enhanced such that it does not require users to play an instrument.

A. Appendix

A.1. User Study – Answers to Questionnaire Sorted by Test Persons

Person A	
Question 1	F.
Question 2	27
Question 3	Amateur/Student at a Local Music School
Question 4	I know basics about tonal improvisation
Question 11	bubble, key, graph
Question 13	In graph mode: when the pitch wraps, i would suggest not connecting the line segments
Question 20	not as easy to follow the guidelines when screen is moving
Question 21	the chord being split if it's on the border
Question 22	wraparound for clarity
Question 39	Bar-Block, Dot-Block, Bar-Linear
Question 43	Learning how to improvise as pianist in a jazz band
Question 44	On Amateur level it depends mostly on the willingness of your teacher whether you get introduced into improvisation or not. The App could help learning on your own.
Question 45	Would mention it if the topic somehow ended up being improvisation
Question 46	Virtual Keyboard could be playable

Person B	
Question 1	M.
Question 2	20
Question 3	Piano lessons as a child
Question 4	None
Question 21	Label specific tone with its pitch (write c4,c5, c6, etc on every C)
Question 22	wrap-around, since its not that hectic
Question 39	Dot-Linear, Bar-Linear, Dot-Block, Bar-Block
Question 42	Block Movement: Little Timer on Bar, so that you know when to change tone
Question 44	Yes, I for example have never learned to improvise at all
Question 45	If I had musically interested friends; yes
Question 46	Advanced Version, where you have to play the accords on the displayed piano or a connected one

Person C	
Question 1	S.
Question 2	29
Question 3	Finished a College Degree
Question 4	I improvise regularly
Question 11	graph, bubble, key
Question 13	keys sind ein bisschen unübersichtlich
Question 22	Graph ist gut, gleichzeitig Freude am Singen haben und Bubbles verfolgen schwer
Question 39	Dot-Linear, Bar-Block, Bar-Linear, Dot-Block
Question 43	In der Mitte(man braucht auf jeden Fall Fachkenntnisse)
Question 44	Zum singen Lernen taugts sicher mega
Question 45	Ja, Hilft womöglich dem einen oder anderen Studenten (z.B. Guidelines kommen echt gut raus)
Question 46	In Verbindung mit Notenmaterial wärs ziemlich nice

Person D	
Question 1	L.
Question 2	23
Question 3	Amateur/Student at a Local Music School
Question 4	I improvise regularly
Question 20	quite confusing to the eye when something's moving quite speedily
Question 21	faster recognition of the respective pitches, more flexibility concerning visualisation
Question 22	wraparound: more suitable for the eye
Question 39	dot-linear
Question 43	when I was 15 / 16 and thought I could conquer the world by stealing blues licks from Clapton
Question 44	the software can definitely contribute to the development of one's auditive understanding and intuition, things which can't be really taught but have to be experienced / discovered
Question 45	I would, for the above reason: you're being guided but the thing that ultimately matters is your intuition
Question 46	As a player of both the piano and the guitar, the visualisation with keys is a bit confusing in the beginning because two different patterns of musical thought are clashing

Person E	
Question 1	S.
Question 2	26
Question 3	Amateur/Student at a Local Music School
Question 4	None
Question 11	bubble, key, graph
Question 20	It moves and it moves quickly, which can be quite distracting. Two suggestions: maybe do not move at all while in a safe area, and maybe decrease acc.
Question 21	On a huge screen, maybe just use a sufficiently large keyboard to eliminate the need for it once the pitch tracker is calibrated initially
Question 22	I prefer the wraparound because it is more "calm" and thus less distracting
Question 39	Dot-Linear, Bar-Linear, Dot-Block, Bar-Block
Question 42	Compared to a regular piano, higher notes are towards the left than the right. Can be slightly confusing at first.
Question 43	When I was good enough to master my instruments for supporting jazz standards. Maybe after 3 years?
Question 44	Yes, that gap is there. Also the software would help fill this gap and would be best accompanied by teaching bits of the relevant theory.
Question 45	Yes, I would if they want to play jazz.

Person F	
Question 1	N.
Question 2	23
Question 3	Student in Higher Musical Education
Question 4	I know basics about tonal improvisation
Question 11	bubble, key, graph
Question 13	The movement of the bubble/ key/ graph system is shaky, maybe it could be at bit softer
Question 39	Dot-Linear, Bar-Linar, Dot-Block, Bar-Block
Question 43	to improve my improvisation
Question 44	Yes there is a gap... maybe you can be lucky and have a good teacher who is showing it to you, but if you don't have one. it's really hard to find good literature/informations about improvisation.
Question 45	I think you have to be interested and open minded for using an app or a computer for practising... but if someone is open to this I would recommend it.

5	6	7	8	9	10	12	14	15	16	17	18	19	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	40	41
4	3	5	5	5	4	5	5	3	5	3	5	5	5	5	4	4	4	5	5	5	3	4	4	5	4	5	4	4	4	5
5	3	5	5	5	4	4	5	5	3	4	4	5	5	5	4	5	5	5	5	5	4	5	5	5	5	1	2	2	5	5
4	5	3	4	5	4	2	5	5	4	5	4	5	1	2	1	5	2	5	5	5	5	5	2	5	5	5	5	5	5	5
2	2	4	4	3	5	4	4	2	2	2	3	3	2	3	4	1	5	5	5	2	3	2	4	4	4	1	1	1	5	5
4	3	5	5	5	5	4	3	1	3	4	5	4	4	5	5	3	5	5	5	5	4	5	5	5	5	4	5	3	5	5
4	3	4	4	4	4	4	4	3	3	5	3	3	5	4	5	3	5	5	4	4	3	3	5	5	4	2	4	2	4	4
2	2	4	4	4	4	5	1	1	4	4	5	4	3	5	4	3	5	5	5	4	5	5	5	5	5	4	5	2	5	5

Figure A.1.

Person G	
Question 1	M.
Question 2	25
Question 3	Finished a College Degree
Question 4	I am used to improvising in front of an audience
Question 11	key,bubble,graph
Question 20	Oktavlagen anzeigen, vllt wie bei MIDI immer bei Ton C
Question 22	Wraparound, ist klarer zu sehen, man muss sich nicht jedes Mal neu darauf einstellen
Question 39	Dot-linear,Bar-linear,Dot-Block,Bar-Block
Question 42	Immer noch: verschiedene Farben für verschiedene Akkordtypen $\rightarrow$ kann man gleich an richtige scale denken
Question 43	Studium, man benötigt schon ein relativ gutes Vorwissen, sonst vllt nach einigen Jahren Jazzerfahrung
Question 44	schwer zu sagen, die App würde aber auf jeden Fall helfen, da man Zusammenhänge zwischen Skalen besser versteht (wie viele Töne ändern sich/wie viele bleiben gleich z.B.)
Question 45	Ja, es nimmt einem in der Phase des Skalenlernens eine Komponente weg: man muss sich nicht komplett selbst erinnern, welche Skalentöne jetzt richtig sind $\rightarrow$ erleichtert das Lernen
Question 46	Besser anzeigen, welche Töne gerade gespielt werden können, ist nicht so gut erkennbar + welche Skala zu welchem Akkord gehört, hierbei können auch die Farben helfen.

A.2. User Study – Questionnaire

Questionnaire: Benedikt Wimmer's Bachelor's Thesis

Welcome to this very important survey. Thank you for filling it all out.

1 About you

1. **Your name**: _____
2. **How old are you?** I am _____ years old.
3. **What kind of degree/musical qualifications do you have?**
   - □ Amateur/Student at a Local Music School
   - □ Student in Higher Musical Education
   - □ Finished a College Degree
   - □ Other: _____
4. **How much experience do you have with improvised music?**
   - □ None
   - □ I know basics about tonal improvisation
   - □ I improvise regularly
   - □ I am used to improvising in front of an audience
   - □ Other: _____

2 Pitch Tracking

Please evaluate the following pitch visualisation methods (not its considering reaction time)

5. **Graph Visualisation** non-intuitive □—□—□—□—□ intuitive
6. **Graph Visualisation** unsightly □—□—□—□—□ visually appealing
7. **Bubble Visualisation** non-intuitive □—□—□—□—□ intuitive
8. **Bubble Visualisation** unsightly □—□—□—□—□ visually appealing
9. **Key Visualisation** non-intuitive □—□—□—□—□ intuitive
10. **Key Visualisation** unsightly □—□—□—□—□ visually appealing
11. **When singing, which methods would you prefer? Answer with an ordering descending from left to right: e.g. (key,graph,bubble) to indicate that you prefer key over graph over bubble**:

    _____
12. **Does the pitch visualisation support/enhance your intuition about musical pitch?**

    not at all □—□—□—□—□ absolutely
13. **If desired, suggest improvements to any of the approaches and/or suggest another approach**:

    _____

    _____

Please evaluate the following approaches for handling out-of-bounds pitches

14. **Follow Pitch with Screen** non-intuitive □—□—□—□—□ intuitive
15. **Follow Pitch with Screen** confusing □—□—□—□—□ clear
16. **Follow Pitch with Screen** unsightly □—□—□—□—□ visually appealing
17. **Transpose Pitch To Fit – "Wraparound"** non-intuitive □—□—□—□—□ intuitive
18. **Transpose Pitch To Fit – "Wraparound"** confusing □—□—□—□—□ clear
19. **Transpose Pitch To Fit – "Wraparound"** unsightly □—□—□—□—□ visually appealing
20. **What confuses you about/could be done to improve "Following the Screen"**:

    _____

    _____
21. **What confuses you about/could be done to improve "Wraparound"**:

    _____

    _____
22. **Which approach do you prefer and – optionally – why?**:

    _____

3 Keyboard Visualisation

Please evaluate the keyboard representation

23. **The piano highlighting ("Show Scale On Piano") makes it easier to see which notes are currently suggested** not at all □—□—□—□—□ absolutely
24. **For this application domain, the "piano roll" is a good alternative to traditional staff notation** not at all □—□—□—□—□ absolutely
25. **I would prefer to be presented with scale suggestions in sheet music notation rather than a "piano roll"** not at all □—□—□—□—□ absolutely

4 Chord Visualisation

Please evaluate the chord representation in general – Turn On "Chord Notes Only"

26. **By closer looking, I can recognize the chord type by the representation's vertical proportions** not at all □—□—□—□—□ absolutely

Please rate the two chord visualisation methods – Turn "Chord Notes Only" Off again!

27. **Dot Visualisation** confusing □—□—□—□—□ clear
28. **Dot Visualisation** unsightly □—□—□—□—□ visually appealing
29. **Dot Visualisation** I fail to perceive horizontal flow □—□—□—□—□ I can easily see horizontal melodic flow
30. **Bar Visualisation** confusing □—□—□—□—□ clear
31. **Bar Visualisation** unsightly □—□—□—□—□ visually appealing
32. **Bar Visualisation** I fail to perceive horizontal flow □—□—□—□—□ I can easily see horizontal melodic flow

Please rate the two approaches towards chord movement

33. **Linear Movement ("Move Chords in Blocks" ist aus)** confusing □—□—□—□—□ clear
34. **Linear Movement ("Move Chords in Blocks" ist aus)** unsightly □—□—□—□—□ visually appealing
35. **Linear Movement ("Move Chords in Blocks" ist aus)** fails to display time-flow □—□—□—□—□ displays time-flow perfectly

36. **Block Movement ("Move Chords in Blocks" ist an)** confusing □—□—□—□—□ clear
37. **Block Movement ("Move Chords in Blocks" ist an)**
unsightly □—□—□—□—□ visually appealing
38. **Block Movement ("Move Chords in Blocks" ist an)**
fails to display time-flow □—□—□—□—□ displays time-flow perfectly

Please rate the guideline visualisation

39. **Please rank the four possible combination of chord visualisation and chord movement to your liking. left to right. descending. e.g. (Dot-Linear, Bar-Block, Dot-Block, Bar-Linear)**:

_____

40. **guideline visualisation** confusing □—□—□—□—□ clear
41. **guideline visualisation** unsightly □—□—□—□—□ musically appealing
42. **Please suggest improvements – if desired – for the featured chord visualisation**:

_____

_____

5 General Questions

43. **In which phase of your musical development would you have made best use of this application?**:

_____

44. **Do you think that there is a gap in today's music pedagogy regarding learning to interpret/improvise over a leadsheet/jazz standard? If so: would the proposed software fill this gap and why?**:

_____

_____

45. **Would you suggest this application to friends/students of yours? If so/If not: why?**:

_____

_____

_____

46. **Room for further suggestions to improve this application/ further general comments**:

_____

_____

_____

List of Figures

Bibliography

[1] J. Aebersold. *Aebersold Vol.2 Nothin' But Blues*. 1981. URL: https://www.jazzbooks.com.

[2] Apple Inc. *Metal | Apple Developer Documentation*. 2019. URL: https://developer.apple.com/documentation/metal.

[3] Y. Broze. *iRb Corpus Released*. retrieved 9/2019. URL: http://www.stacoscimus.com/irb-corpus-released/.

[4] M. Claypool and D. Finkel. "The effects of latency on player performance in cloud-based games." In: *2014 13th Annual Workshop on Network and Systems Support for Games*. Dec. 2014, pp. 1–6. DOI: 10.1109/NetGames.2014.7008964.

[5] J. Driedger and M. Müller. "A Review of Time-Scale Modification of Music Signals." In: *Applied Sciences* 6.2 (Feb. 2016), p. 57. ISSN: 2076-3417. DOI: 10.3390/app6020057. URL: http://dx.doi.org/10.3390/app6020057.

[6] V. Eremenko, E. Demirel, B. Bozkurt, and X. Serra. *JAAH: Audio-aligned jazz harmony dataset*. June 2018. DOI: 10.5281/zenodo.1290737. URL: https://mtg.github.io/JAAH/.

[7] G. Gokul, Y. Yan, K. Dantu, S. Y. Ko, and L. Ziarek. "Real Time Sound Processing on Android." In: *Proceedings of the 14th International Workshop on Java Technologies for Real-Time and Embedded Systems*. JTRES '16. Lugano, Switzerland: ACM, 2016, 3:1–3:10. ISBN: 978-1-4503-4800-3. DOI: 10.1145/2990509.2990512. URL: http://doi.acm.org.eaccess.ub.tum.de/10.1145/2990509.2990512.

[8] Hal Leonard Coroporation. *The Real Book*. URL: https://officialrealbook.com/history/.

[9] Jamey Aebersold Jazz. *Introduction To The Scale Syllabus*. 2010. URL: https://www.jazzbooks.com/mm5/download/FREE-scale-syllabus.pdf.

[10] Jamey Aebersold Jazz. *Jamey Aebersold Biography*. 2019. URL: https://www.jazzbooks.com/mm5/merchant.mvc?Screen=JBIO&Store_Code=JAJAZZ.

[11] Jamey Aebersold Jazz. *Nomenclature*. 2010. URL: https://www.jazzbooks.com/mm5/download/FREE-nomenclature.pdf.

[12] Khronos Group. *OpenGL Overview*. 2019. URL: https://www.opengl.org/about/.

[13] L. von Köchel. *Nachtrag zum Chronologisch-thematischen Verzeichniss sämmtlicher Tonwerke Wolfgang Amade Mozart's*. 1862. URL: https://books.google.de/books?id=kV4VAAAAYAAJ&pg=PA409&hl=de&source=gbs_selected_pages&cad=2#v=onepage&q&f=false.

[14] E. Larson and S. Errede. *Real-Time Time Domain Pitch Tracking Using Wavelets*. Jan. 2006. URL: https://www.scribd.com/document/59212684/Real-Time-Time-Domain-Pitch-Tracking-Using-Wavelets.

[15] B. Laugwitz, T. Held, and M. Schrepp. "Construction and Evaluation of a User Experience Questionnaire." In: vol. 5298. Nov. 2008, pp. 63–76. DOI: 10.1007/978-3-540-89350-9_6.

[16] M. Levine. *Das Jazz Theorie Buch*. Advance Music, 1996.

[17] G. Loy. "Musicians Make a Standard: The MIDI Phenomenon." In: *Computer Music Journal* 9.4 (1985), pp. 8–26. DOI: 10.2307/3679619. URL: https://www.jstor.org/stable/3679619.

[18] A. Malcolm and J. Mitchell. *A treatise of musick, speculative, practical and historical*. 1721. URL: https://archive.org/details/treatiseofmusick00malc/page/116.

[19] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999. URL: https://nlp.stanford.edu/fsnlp/promo/.

[20] MusiClock. *App | MusiClock*. 2018. URL: https://www.getmusiclock.com/app-1.

[21] C. Pratt. "The Spatial Character of High and Low Tones." In: *Journal of Experimental Psychology* 13 (1930), pp. 278–285. URL: http://www.aruffo.com/eartraining/research/articles/pratt30.htm.

[22] F. Sikora. *Neue Jazz-Harmonielehre*. Schott Musik International, 2003. URL: https://de.schott-music.com/shop/neue-jazz-harmonielehre-no91932.html.

[23] Superpowered Inc. *iOS and Android Latency Test App*. 2019. URL: https://superpowered.com/latency.

[24] Synthesia LLC. *Synthesia*. Version 10.5.1. Jan. 29, 2019. URL: https://synthesiagame.com.

[25] 1.-1. Taylor Jane. *Twinkle, twinkle, little star*. First edition. New York : Little, Brown, 2011., 2011. URL: https://search.library.wisc.edu/catalog/9910112130302121.

[26] technimo. *iReal Pro - Music Book and Backing Tracks*. 2019. URL: https://irealpro.com.

[27] The Jackson Five. *ABC*. 1970. URL: https://www.discogs.com/de/The-Jackson-5-ABC/release/5976514.

[28] S. Torrance and F. Schumann. "The spur of the moment: what jazz improvisation tells cognitive science." In: *AI & SOCIETY* 34.2 (June 2019), pp. 251–268. ISSN: 1435-5655. DOI: 10.1007/s00146-018-0838-4. URL: https://doi.org/10.1007/s00146-018-0838-4.

[29] T. Yamaoka. *Vocal Pitch Monitor*. 2019. URL: https://apps.apple.com/app/vocal-pitch-monitor/id842218231.