GuitarPie: Using the Fretboard of an Electric Guitar for Audio-Based Pie Menu Interaction

Frank Heyen
VISUS, University of Stuttgart
Stuttgart, Germany
frank.heyen@visus.uni-stuttgart.de

Marius Labudda
VISUS, University of Stuttgart
Stuttgart, Germany
marius-labudda@hotmail.com

Michael Sedlmair
VISUS, University of Stuttgart
Stuttgart, Germany
michael.sedlmair@visus.uni-stuttgart.de

Andreas Fender
VISUS, University of Stuttgart
Stuttgart, Germany
andreas.fender@visus.uni-stuttgart.de

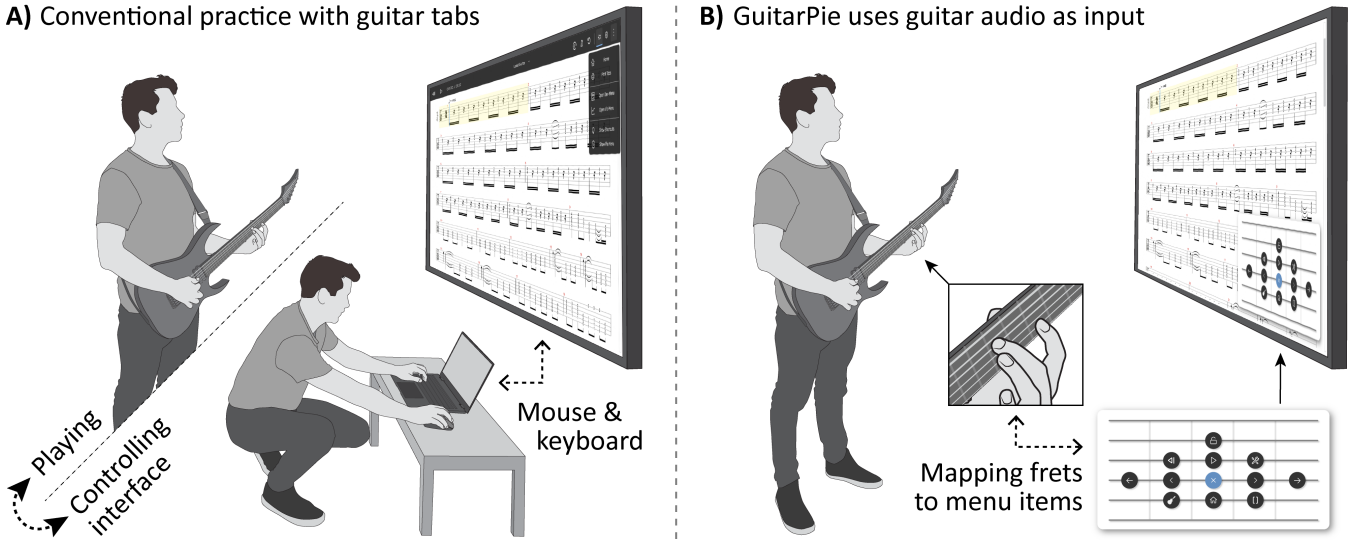

Figure 1: A) Guitar practice with digital tabs involves frequent switches between playing and interface control. B) GuitarPie maps menu items to guitar frets akin to a pie menu, enabling audio-based interface control without letting go of the guitar.

Abstract

Nowadays, electric guitars are often used together with digital interfaces. For instance, tablature applications can support guitar practice by rendering and playing back the tabs of individual instrument tracks of a song (guitar, drums, etc.). However, those interfaces are typically controlled via mouse and keyboard or via touch input. This means that controlling and configuring playback during practice can lead to high switching costs, as learners often need to switch between playing and interface control. In this paper, we explore the use of audio input from an unmodified electric guitar to enable interface control without letting go of the guitar. We present GuitarPie, an audio-based pie menu interaction method. GuitarPie utilizes the grid-like structure of a fretboard to spatially represent audio-controlled operations, avoiding the need to memorize note sequences. Furthermore, we implemented TabCtrl, a tablature interface that uses GuitarPie and other audio-based interaction methods for interface control.

CCS Concepts

• **Human-centered computing** → **Sound-based input / output**; *Interaction techniques.*

Keywords

Music practice, Tablature interfaces, Radial menus, Marking menus

ACM Reference Format:
Frank Heyen, Marius Labudda, Michael Sedlmair, and Andreas Fender. 2025. GuitarPie: Using the Fretboard of an Electric Guitar for Audio-Based Pie Menu Interaction. In *The 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25), September 28-October 1, 2025, Busan, Republic of Korea.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3746059.3747799

UIST '25, September 28-October 1, 2025, Busan, Republic of Korea

ACM ISBN 979-8-4007-2037-6/2025/09
https://doi.org/10.1145/3746059.3747799

1 Introduction

Musical instruments such as the electric guitar are increasingly used together with digital interfaces. In particular, digital tools and online tutorial platforms can lower the entry barrier for learning to play the guitar as a self-taught hobbyist, between guitar lessons, or during band practice. For instance, interfaces that interactively render and play back guitar tablatures (or guitar *tabs*), such as *Guitar Pro*[1] or *Ultimate Guitar*[2], facilitate practicing with the electric guitar through a beginner-friendly alternative to sheet music. In addition, digital tablature interfaces typically support interactive playback of synthesized versions of the practiced song (or synchronized with the song's original audio). Importantly, such applications support many functionalities for navigation and for configuring playback, including muting and unmuting individual audio tracks (e.g., the drum track to be played alongside when practicing, while the guitar track is muted), jumping to specific positions, or choosing a different song. However, operating those interfaces while holding the guitar can be challenging [5]. Concretely, an electric guitar occupies both hands and affords an upright posture. Constantly switching between playing the guitar and operating a desktop PC or a small tablet can be disruptive. The mouse and keyboard might even be hard to access, e.g., when practicing with multiple people, using a shared large screen. This means that, depending on the practicing setup, the switching costs can become even higher.

Figure 1 (A) shows an example setup with tabs on a large screen (resembling practice using a TV screen for instance). However, the problem of switching between playing and interface control also occurs in other setups [6, pg. 4]. A desktop environment forces the guitarist to sit close to the mouse and keyboard when trying to reduce switching costs. A mobile tablet with touch input is more flexible as it can be moved around and placed suitably. However, its smaller size limits the input and output surface for the tabs, especially when practicing together with others. Our goal is hence to retain a high level of control, while enabling flexible practicing setups. Physically augmenting the guitar with custom hardware or using extra equipment can partially achieve this goal, but with reduced flexibility. Concretely, using such inputs might be disruptive and even separate equipment, such as foot pedals, can be inconvenient [5, Sect. 3.1.2].

To enable interface control while avoiding physical modifications and extra equipment, this work explores the use of the *audio* from the guitar itself as a means of menu interaction and as an input method for controlling tablature interfaces (Figure 1 B). Based on the insights of a design workshop, we designed the **GuitarPie** technique, which leverages the grid-like structure on a guitar fretboard to map audio signals to menu item locations. At the same time, GuitarPie uses *relative* fret positions and can hence be used almost anywhere along the fretboard, similar to how conventional pie menus open around the current cursor position [10]. Consequently, instead of having to memorize specific note sequences for each operation, those sequences for triggering operations emerge implicitly through easy-to-visualize spatial relationships on the fretboard. We implemented a specific instance of GuitarPie in our tablature interface application **TabCtrl**. In addition to GuitarPie, our TabCtrl application integrates other audio input methods for specific tasks, like playing a snippet of a song to open that song. We explore operations at different levels of granularity, ranging from starting playback to lower-level operations, such as moving the playhead by a single bar as well as marking regions. With this, TabCtrl enables audio-based tabs navigation, configuration, and song switching during practice. Our contribution is twofold:

- Our **GuitarPie technique** is inspired by pie menus and only requires audio inputs from an unmodified electric guitar. We describe the design of our technique, which is also informed by the outcome of a design workshop.
- Our **TabCtrl application** uses an instance of GuitarPie as the main means of interface control. Furthermore, TabCtrl utilizes several additional audio processing methods, such as matching parts of a song, to further reduce the need to switch to mouse and keyboard input.

2 Related Work

The two main aspects of our work, i.e., guitar-based menu design and its application within guitar tablature interface control, build on previous research in audio-based menu interaction, audio processing algorithms, and musical instrument-based interface control. In this section, we focus on literature that is closely related to our main contribution. We will discuss the broader guitar learning literature and complementary approaches in Section 8.

2.1 Controlling Tablature Interfaces

An overarching goal of our research is to enable controlling a guitar tablature interface without letting go of the guitar and without requiring to extend the guitar's hardware. Martinez Avila mentions a prototype that uses MIDI input for controlling media playback [36, pg. 108]. However, the input method is limited to explicit note sequences to be looked up or memorized and lacks abilities for fast, fine-grained interface control. Another possible input modality is voice input, which can potentially be complementary to our approach, and can be useful as an additional option for high-level commands, like switching to a specific song. For instance, the commercial tablature interface *Guitar Flow*[3] supports voice commands for many of its interface operations. However, while voice input can be useful for some interactions, it comes with typical limitations of speech interfaces, such as willingness of users to use their voice as an input method [5, Sect. 3.1.3].

2.2 Non-verbal Audio-Based Menu Interaction

Not only voice commands but also non-verbal audio input, such as humming, can be used as input for menu interaction [11]. This type of input is also relevant to our GuitarPie technique, due to similar approaches and challenges in terms of interaction design. Non-verbal audio input has been used for accessibility [41, 45] or for use cases in which both hands are occupied (e.g., when driving [19]). As opposed to voice input, non-verbal sounds typically have no inherent semantics and hence require careful interaction design to make them suitable for menu interaction. This type of input can either be used as a complementary modality, e.g., for

[1] https://www.guitar-pro.com (accessed: 12th of July, 2025)
[2] https://www.ultimate-guitar.com (accessed: 12th of July, 2025)
[3] https://guitarflow.io (accessed: 12th of July, 2025)

triggering clicks [54] or as the single modality for menu interaction. For instance, Zinck and Vogel [55] evaluate humming as a modality for menu control. They map the pitch of the humming sound to menu items (arranged on a circle with the continuous pitch mapped to the angle). In addition, sequences of humming notes enable hierarchical menus.

Our technique also uses pitch as input, but is designed around the use with an electric guitar. Concretely, instead of proportionally mapping the pitch value to a linear list, the menu items of our GuitarPie technique are spatially arranged to match relative locations on a fretboard—thereby mapping the graphical representation with physical fret positions.

2.3 Instrument Audio Processing

Our concept and prototype relies on algorithms and approaches within the music informatics literature—in particular within processing audio of musical instruments. Many researchers developed approaches for transforming the raw audio waves into a format that makes it easier to interpret within the context of music. We describe key approaches in the following.

2.3.1 Detecting Musical Features from Raw Audio. Automatic music transcription (**AMT**) takes a raw audio stream as input and outputs symbolic representations along the time axis (e.g., the detected musical notes in MIDI format). **Monophonic** AMT approaches assume that a single note is played at a time. For instance, *CREPE* [28] is a monophonic library that has been used in many research projects since its release in 2018. **Polyphonic** approaches can detect multiple notes played at once (e.g., chords). Polyphonic automatic music transcription is based on multi-pitch estimation algorithms, which detect multiple fundamental frequencies of an audio signal [12]. In recent years, the uptake of deep learning techniques led to many robust monophonic and polyphonic AMT approaches, enabling low-latency real-time transcription [9], even with the presence of background instruments [24]. Researchers also developed models that were specialized for specific instruments, such as the piano [30]. Besides AMT for the piano, researchers and practitioners have also investigated guitar tablature transcription [13, 42]. *MIDI Guitar* by Jam Origin[4] is a commercial tool that supports real-time conversion from clean guitar audio to MIDI notes. In addition, researchers investigated instrument-agnostic approaches [53]. *Basic Pitch* [8] is a polyphonic library that estimates pitches of multiple instruments within a given audio snippet.

2.3.2 Matching, Classification, and Score Following. A general approach for matching audio sources is to use *Dynamic Time Warping* (**DTW**, [40]) or variations thereof on the audio signal. This is useful when there are slight variations in tempo, but not if the recordings differ in other ways (different audio qualities, different recording setups, etc.). Therefore, researchers have developed approaches for specifically matching instrument audio, as those can utilize musical features and metrics. A common way is to use the previously discussed pitch estimation approaches as a first step, i.e., first extracting pitches of both recordings based on the used note system (e.g., the 12 notes of the Western note system) in order to then compare those instead of the raw signals.

[4]https://www.jamorigin.com (accessed: 12th of July, 2025)

Matching two instrument audio sources has previously been researched extensively [1, 15, 16, 26, 47]. A typical use case for audio matching is **score following** [3, 22]. In the case of the piano for instance, score following enables automatic sheet turning while playing [2, 4, 27, 34, 43]. Analogously, matching and score following approaches have been developed for the guitar [18, 20, 33, 46] or other string instruments, such as the shamisen [21]. Relatedly, *Soloist* by Wang et al. features *region-querying* in guitar tutorial videos [52, pg. 8], i.e., using matching to jump to a part of a tutorial (instead of a song). However, the other parts of their interface are controlled via mouse and keyboard, as they focus on improving practice with tutorial videos (extracting musical parts and estimating the learner's performance) and not on full interface control via guitar audio.

Modern guitar tracks (especially background guitars) typically contain many more repetitions than classical piano pieces, which is why playing a specific riff is generally not enough to move the playhead to a well-defined unique position, as the same riff typically appears multiple times. While audio matching can be highly effective for songs with unique parts, we argue that for typical guitar tracks (i.e., as part of pop or rock songs), fine-grained explicit operations are required to fully support the functionalities of today's guitar tablature interfaces. Therefore, in our TabCtrl application, we use our GuitarPie technique as the main method for interface control, and utilize audio matching *exclusively* for specific tasks, such as switching to specific songs (as a substitute for selection from a list or text entry).

3 Design Workshop

In this section we describe our design workshop with guitarists about audio-based interaction, which led to GuitarPie's design. The workshop was approved by the ethics committee of our institution.

3.1 Workshop Participants

We recruited five guitarists with different levels of expertise through convenience sampling. The five participants were male and between 30 and 40 years old. They self-rated their expertise with the guitar as beginner (2 participants) and intermediate (3). They have been playing actively for three (2), four (1), and ten (2) years. When active, they play for about one to two hours (3), and nine hours (1) per week. Tablature interfaces are used sometimes by two and often by three participants during practice. Sheet music is used rarely (2), never (1), sometimes (1), or often (1). Three participants often practice together with others, while two do so never or rarely. Common practice setups (multiple answers possible) were a PC or laptop on a desk (3), followed by a tablet (2) or printed tab (2). A screen without a desk (e.g., TV setup), laptop on the couch, and phone were each only chosen once as an answer. All participants used digital tablature interfaces before, such as *Guitar Pro* (4) and *Songsterr* (4). One participant used the computer game *Rocksmith* [49].

3.2 Workshop Procedure

First, the participants were greeted and each of them signed a consent form. The two-hour workshop consisted of two parts: 1) reflecting on current usage and problems with tablature interfaces, and 2) coming up with possible solutions for using guitar audio

for interaction. In each part, participants first spent time alone to think and take notes, to then discuss in the group. Participants sat around a table and used sticky notes to collect and arrange their ideas. Throughout the workshop, guitars were available for bodystorming, i.e., trying out different interactions.

3.3 Workshop Results

After transcribing the video recording, we extracted the most insightful ideas and structured them in an affinity diagram (Figure 7 in the appendix). In the following, we summarize our findings.

3.3.1 Previous Experiences with Tablature Interfaces. We asked participants about their previous experiences and "pain points". At this point during the workshop, we did not introduce our goal of using audio input yet, and encouraged participants to focus purely on their previous experiences.

Regarding pain points, the participants mentioned that they often lack overview of the song structure especially with multiple instruments and that navigating takes a lot of mouse movement: Displaying a lot of information at once requires them to zoom in and out or scroll often, for example when jumping back and forth between different sections to practice them. Switching between or selecting multiple instrument tracks was also seen as annoying. A general pain point is the frequent switch between playing the guitar and using the interface, which often requires putting away the guitar and the pick (plectrum) and picking both up again, followed by "*homing*", that is, putting the fretting hand in the position required for playing the current part.

3.3.2 Audio Gesture Elicitation. At the start of the second part of the workshop, we introduced our goal of using guitar audio for interacting with the tablature display. We clarified that we only look for approaches that do not modify or augment the guitar itself (i.e., no custom hardware or equipment that needs to be attached to or built into the guitar) and that do not require other equipment such as a USB pedal. Furthermore, we encouraged the participants to think from just the user's perspective and *not* consider technical feasibility aspects, such as implementation or limitations.

The participants came up with interactions for specific actions, such as simply playing a part of a song to jump to this song, the specific instrument track, or even to the exact time (i.e., score following). Another idea was to use sliding along a string as directional input: sliding up makes a sound with increasing pitch that would be mapped to an action that increases a value like the zoom level, sliding down would do the opposite.

The participants thought about ways to differentiate between playing and interaction. Having a separate interaction mode would allow to use any input for control, even if it is identical or similar to something that occurs in a song. One notable example was to use two audio signals, one from the guitar cable and one from the device's microphone, and only process the audio input whenever the guitar is turned down with its gain knob, i.e., when only the microphone picks up sound. Another idea was to create sounds that one would never produce while playing, e.g., by strumming the strings on the guitar's head, atop the nut. They also discussed audio gestures that are potentially distinct enough from typical audio while playing. These include pressing strings directly onto the pickup, hitting/slapping strings, or playing a specific pattern.

Throughout the discussion, the participants mentioned different kinds of requirements. For flexibility, the interface should support multiple actions in a similar way, e.g., allowing to select something from a list instead of having a separate command for each item. Since every guitarist has different skills and their own taste, commands should be user-customizable. Ideally, the interface would support multiple users, so everyone can control the interface in a band setting. The participants emphasized the importance of the ability to start interacting (almost) anywhere on the guitar to avoid having to move away from the current playing position and back. In addition, the sounds produced during interaction should not sound annoying. For an easier mental model, a single command should not have a repeating pattern — repeated inputs should instead map to repeated actions. Every interaction should be easy to learn and remember (avoiding complex memorized note sequences).

While sharing ideas, an ensuing discussion emerged about more abstract interactions. For example, instead of remembering many audio gestures, actions could be selected from a list with lower-level inputs such as *up*, *down*, and *confirm*. Inspired by a game controller, one participant suggested that a few notes next to each other on the guitar could be used for navigation using a *relative* interaction: After playing a first note which serves as the center, the note to the left of it maps to "move left", and so on. The participants then noted that besides these navigation directions for menu items or moving around the song, other notes could serve as 'buttons' for features like play/pause. More complex sequences of interactions, such as selecting a start and end of a region and then activating looping and playback, could then be supported via sub-menus. To help the user navigate without remembering the menu, the participants desired a visual representation of the current menu state on the screen. This overall 'relative interaction' approach reminded us of pie menus. All of this ultimately converged into the main technique (GuitarPie) and inspired the terminology of this work.

At the end (not as part of the main brainstorms), we also inquired how our participants felt about using voice controls as an alternative to using guitar audio. One participant pondered using singing, which would keep the interaction musical. Even though the participants saw benefits of voice control such as ease of use and intuitiveness, they were largely skeptical — arguing that while voice controls have existed for years in other applications, they never use them. One participant even "*despises voice commands*" and finds it uncomfortable to talk to a computer. This sentiment with an overall hesitancy towards voice input is similar to the workshop results of *Stretchy Strap* [5, Sect. 3.1.3]. Yet, some participants could imagine using voice commands, but exclusively for complex commands that are effortful to perform but easy to say, such as "*select the chorus and loop it*".

3.4 Summary and Requirements

In sum, we collected the following insights and requirements for audio-based input techniques from our design workshop:

- Support for the following common actions: play/pause, instrument track selection, marking a region and looping it, navigating between sections, and adjusting speed and zoom.

- Separate play and control modes.
- Control interface without having to move the fretting hand away too far from the current playing position.
- Having a visual/spatial representation (instead of memorizing notes).
- General usability requirements: easy-to-learn and intuitive.
- Supporting user-customizable commands.

Based on these requirements, we designed the GuitarPie technique and applied it in a tablature interface control context by implementing TabCtrl. In the subsequent sections, we describe our technique and application, followed by our design rationale.

4 GuitarPie and TabCtrl

In this section, we describe our TabCtrl application, a tablature interface, which contains a specific instance of our GuitarPie technique plus other audio-based input methods. A walkthrough of GuitarPie and TabCtrl can be seen in Video Figure A in the supplemental materials. In terms of look and functionalities, TabCtrl resembles a conventional tablature application (Figure 2) and can be fully controlled with a touch screen or mouse and keyboard. Specifically, a menu bar at the top provides access to all functionalities with the mouse. However, as opposed to conventional tablature interfaces, crucial functionalities can also be controlled via guitar audio without using the menu bar. We elaborate in the following.

4.1 Song Selection

The application starts in the *Song Selection* (Figure 2 A), in which songs can be selected with the mouse and keyboard. Furthermore, for each song, the learner can record one or more short snippets that the learner associates with that song (for instance the intro riff or a key part of the solo), which are then stored as *song bookmarks*. Learners can directly open their desired song to practice simply by playing the bookmark snippet, i.e., while already holding the guitar without typing the song name or selecting it from the list.

4.2 Tabs Player

Once a song was selected or a bookmark snippet was played, the application transitions to the *Tabs Player* state (Figure 2 B), which uses our GuitarPie technique as the primary way of navigating and controlling the interface.

4.2.1 Menu Items of GuitarPie Instance. When playing a note on the fourth string (D string in standard tuning), the main menu of GuitarPie (Figure 3 top) appears in the lower-right corner of the screen (Figure 1 B), representing a part of the fretboard. We refer to the position on the fretboard where the menu was opened as the *center* (blue X-symbol in Figure 3 top). In the following, we describe the TabCtrl-specific menu items and sub menus.

- **Close menu** (center). Playing the same fret that opened the menu closes it.
- **Play** (up). Starts playback at the current playhead position and closes the menu.
- **Stepwise navigation** (center row left and right). Playing a fret left and right next to the center moves the playhead forward or backward by one bar, respectively. The frets that are a full step down or up (i.e., two frets to the left or right) move the playhead to the previous and next section, respectively.
- **Back to start** (up-left). Moves the playhead to the song's start.
- **Home menu** (down). Returns to the *Song Selection*. This means that learners can swiftly switch songs without mouse and keyboard simply by returning to home and immediately playing the bookmark snippet of the desired song.
- **Lock** (two up, above menu). Closes the menu and disables it. This is for cases in which the learner wants to practice a part without playing it back. The menu can only be enabled again via mouse or via unlock audio commands (details below in Section 4.2.2).

The remaining menu items open sub menus (Figure 3 s1–s3) with the played fret as the new center, similar to traversing hierarchies in marking menus.

- **Settings** (up-right). Opens the *Settings* sub menu (Figure 3 s1) with the following operations relative to the *Settings* fret.
  - **Toggle Count-in** (up).
  - **Metronome on/off** (down).
  - **Zoom out/in** (up-left). Makes the tablature smaller or larger.
  - **Tempo decrease/increase** (left).
  - **Volume down/up** (down-left).
- **Tracks** (down-left). Opens the *Tracks* sub menu (Figure 3 s2). Allows to switch tracks via up and down items (typically, tablature interfaces separate the guitar tracks, drum track, etc. of a song, with separate tabs for each). Furthermore, tracks can be muted or set to solo playback.

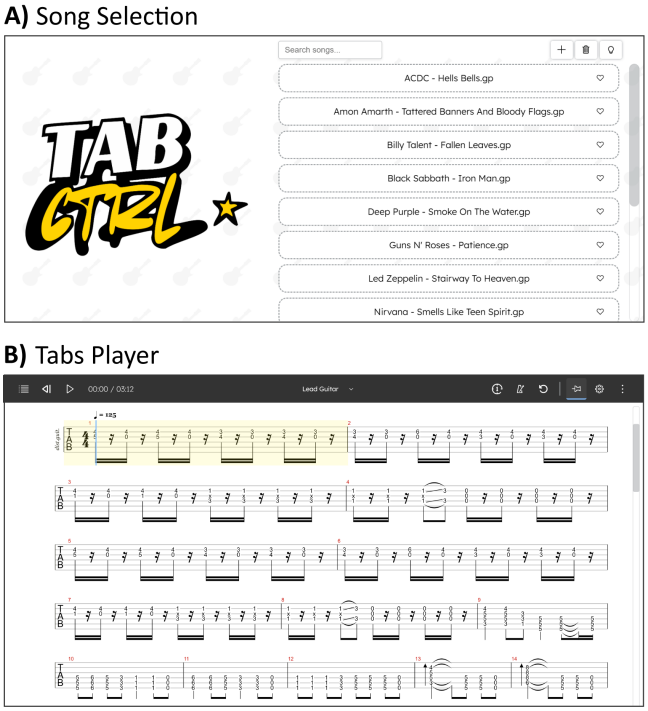

Figure 2: The two states of TabCtrl. A) In the *Song Selection*, the learner can select a song via mouse or by playing a pre-recorded song bookmark. B) The *Tabs Player* resembles an ordinary tabs interface, but can also be controlled via audio.

- **Region** (down-right). Opens the *Region* sub menu (Figure 3 s3). Allows to mark the bar of the current playhead position as the beginning or end of a region, and to remove the region mark. This sub menu also contains an item for toggling looped playback. If looping is on, then the region is looped when playing (or the whole song if no region was specified). If looping is off, then the playback ends at the end of the region or the song, respectively.

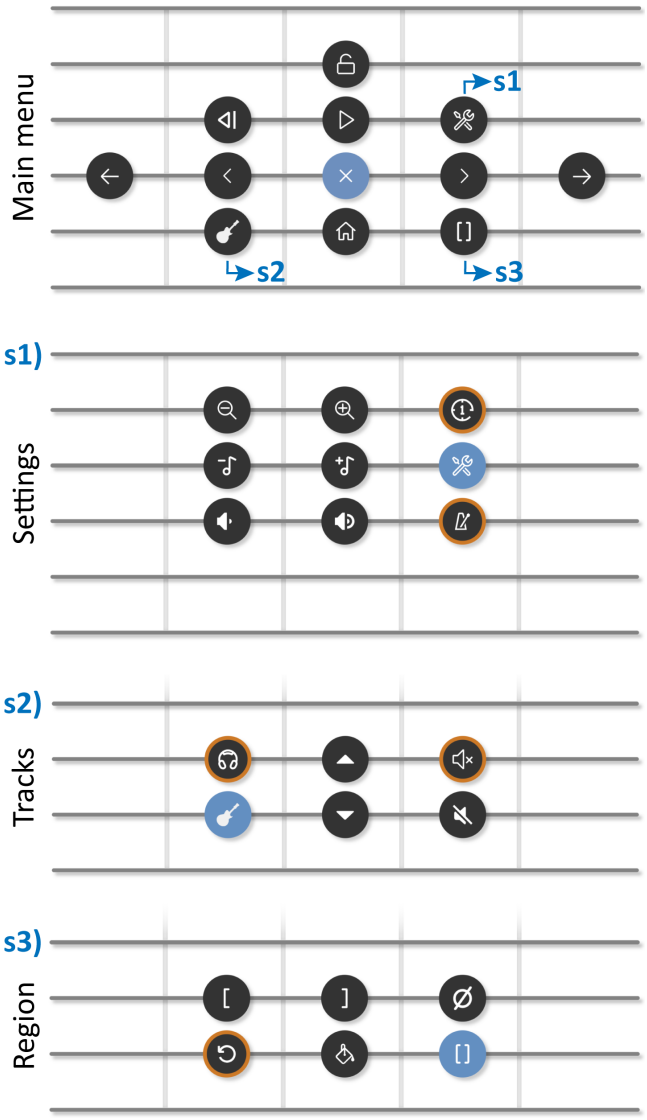

Figure 3: Hitting a note on the fourth string in TabCtrl opens a hierarchical GuitarPie menu (top) with several operations and sub menus (s1–s3).

4.2.2 Stopping Playback and Unlocking Menu. During playback, i.e., while using the guitar for playing and practicing the song, GuitarPie is inactive. This is to avoid ambiguities while practicing the song versus using the menu. Many functionalities such as settings are not required while playing and we hence strictly distinguish between playing the song and operating the interface. The menu can also be locked explicitly (menu item above *Play*). For stopping playback and for unlocking the menu, another type of audio input is required (as GuitarPie is inactive). Per default, TabCtrl stops playback when the learner stops playing the song in non-silent parts of the active track (this option can be toggled on or off). In addition, the learner can define unique *audio commands* (snippets of audio waves) that stop the playback and unlock the menu explicitly via audio. Those audio commands can be sequences of notes that would not be played together in a song (e.g., a brief melody that is not part of the songs that the learner is practicing). Alternatively, the learner can simply turn on the option to stop playback by hitting a single very high note that is rarely used in songs (fret 20 or higher on the 1st string). While not all options for pausing are applicable in isolation for all songs (e.g., silent parts of the guitar track or songs that contain very high notes), we aim to cover most use cases through those complementary options.

4.3 Audio Setup Options

There are multiple options for capturing the guitar's audio (also see Video Figure B). One possible setup is to simply use a microphone close to the guitar amplifier. Alternatively, the learner can use an audio splitter to connect the guitar output to both, the PC (via audio interface) as well as to the amplifier. This way, the PC can always receive the clean guitar audio regardless of distortion and effect settings. More generally, connecting the guitar audio directly (instead of using a microphone) avoids capturing undesired environmental sounds and is overall less susceptible to noise and variations in capture quality. If the learner has a MIDI pickup[5], then this device can be used as an alternative input for controlling the menu. However, optional commands based on raw audio recordings (see Section 4.2.2) are not supported when using only a MIDI pickup. We discuss further trade-offs of those setups from an implementation perspective in Section 6.1.

5 Design Rationale

The goal of our design is to balance aspects like discoverability, memorability, hardware requirements, and robustness. In this section, we describe the design rationale of our GuitarPie technique.

5.1 Pie Menu Metaphor

GuitarPie is inspired by pie menus [10]. Analogous to appearing around a cursor position (in the case of conventional pie menus), GuitarPie uses the first played position on the fretboard as its center, with all menu items surrounding it. There are two major reasons for this decision (compared to using a fixed fret). First, due to the length of the fretboard, it would typically require looking at the fretting hand to find a specific fret. In contrast, each of the six strings can be found more easily without looking. Second, during practice, different riffs are played at varying locations along the fretboard. Therefore, the fretting hand does not need to be moved considerably when switching between playing and using the menu, which was one of the requirements from our design workshop.

[5]Note that we consider a MIDI pickup to be additional equipment. We mention a MIDI pickup for the sake of completeness as it is inherently supported without major changes to our source code.

5.2 Spatial Fretboard Layout Versus Audio

Our main goal is to *not* require additional sensing hardware such as cameras, which means that we do not know the actual finger position. To infer in which direction on the fretboard the user's finger went after opening the pie menu, we only need to know the difference between the first played note (to open the menu) and the follow-up note, while taking the relative note offsets of the guitar strings into account. Having this unique mix of a spatial representation for the learner (instead of memorizing notes) while actually using audio as the input method requires considerations for aspects such as ambiguities and different guitar tunings.

5.2.1 Constraints of Repeating Notes on the Fretboard. The guitar contains ambiguities by design. In particular, most notes can be played on more than one string. Therefore, menu items cannot be placed arbitrarily when designing GuitarPie menus. For instance, if there is an item on the fourth string, then there cannot be another item five frets down on the third string, i.e., there is a maximum of four to five items per string. Furthermore, we need to assume that the learner uses the fourth string to open the menu as we cannot check from the audio whether the guitarist indeed used the correct string for opening the menu.

5.2.2 Tuning-Agnostic Pie Menu. Many songs do not use the standard guitar tuning (e.g., some songs use tunings in which *all* strings are tuned a half step or full step down from standard tuning). For instance, when switching to a song that would require to change the tune of all strings equally, the learner might not tune the guitar accordingly as the song still sounds correct for practicing purposes (just pitch-shifted up or down). Therefore, the tuning stored in the tabs file does not necessarily indicate the actual tuning of the guitar. GuitarPie typically works without adjustments if all strings are equally pitch-shifted, as the frequency differences relative to the center of the pie menu are still the same. Similarly, using a capo does not require adjustments to the technique. In any case, the learner needs to open the menu at locations with enough space around it on the fretboard (third fret or higher relative to the capo). However, if there are relative tuning differences between the strings (e.g., switching between standard tuning and Drop D), then the menu would not work anymore on the sixth string without knowing the tuning (e.g., some standard tuning songs can be played in Drop D and sound the same). Hence, to be agnostic to dropped tunings (Drop C, Drop D), we do not assign any operations of the pie menu to the sixth string. Relatedly, due to the tuning-agnostic design, we do not display the string names or fret numbers around the GuitarPie menu, meaning that the center (the fret that was played) always acts as the visual anchor when using the menu.

6 Implementation

The dataflow of TabCtrl for both system states (*Song Selection* and *Tabs Player*) can be seen in Figure 4. In our reference implementation, we used HTML/JavaScript for the frontend (`Selection UI`, `Tabs UI`, and `GuitarPie menu` in Figure 4) and Python for the backend (all other boxes in Figure 4). The backend and the frontend communicate via HTTP and WebSockets. The tabs are rendered and played back using *alphaTab*[6].

[6]https://alphatab.net (accessed: 12th of July, 2025)

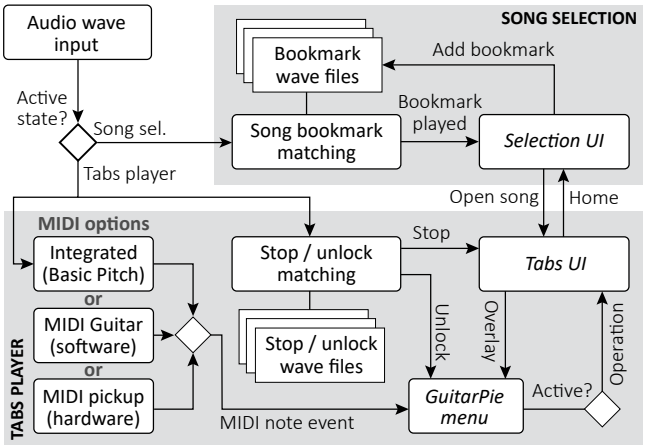

Figure 4: Overview of TabCtrl's dataflow. The options *MIDI Guitar* and *MIDI pickup* are external software and hardware, respectively. Rounded boxes are processing steps (regular font) or parts of the frontend (italics). Arrows indicate data or events. Stacked boxes indicate files.

6.1 MIDI Source

As described in Section 4.3, there are different options for picking up audio from the guitar. From an implementation perspective, there are three options for the MIDI input source (see 'MIDI options' in Figure 4). The first option is to use an actual MIDI device, i.e., a physical MIDI pickup add-on for the guitar. This provides good performance and connecting the guitar's raw audio is not required. However, this is subject to the availability of extra hardware. Hence, we envision this for cases where a MIDI pickup is installed anyway. The second option is to use software that converts a clean guitar audio signal into MIDI notes. For instance, *MIDI Guitar* (footnote 4) emulates a MIDI device by processing guitar audio. Its audio processing is specialized for a clean guitar input and optimized for low latency. However, the latency requirements of GuitarPie are not as strict as in *MIDI Guitar*'s use case of real-time synthesizers operated with the guitar. Therefore, we integrated an approach that uses a larger time window in the audio input, but adds flexibility. Concretely, we use *Basic Pitch*[7] for converting the real-time audio input to MIDI notes (`Integrated` in Figure 4). This does not require external software or hardware and even allows for setups that pick up the amplifier's audio with a microphone, in case the user does not have an audio splitter and an audio interface.

A hardware MIDI pickup is generally quite reliable for our purpose, so that it does not require additional processing. When using external software MIDI inputs, we only use the most likely note if multiple notes arrive in quick succession. Integrated MIDI conversion, i.e., our use of *Basic Pitch*, required additional adjustments in order to be used as a robust input for GuitarPie. We describe those adjustments in the following.

6.1.1 Basic Pitch Adjustments for Live Audio. MIDI note prediction algorithms such as *CREPE* [29] and *Basic Pitch* [8] are normally used for audio recordings with a well-defined start and end of an

[7]https://github.com/spotify/basic-pitch (accessed: 12th of July, 2025)

audio file. However, in real-time audio processing, the operating system simply sends audio samples in blocks (generally with a fixed time window). In our application, however, learners should be able to play notes at any time without explicitly starting a recording. Therefore, we need to make a few considerations for our usage of *Basic Pitch*, so that it works in real-time and without explicitly starting and stopping an audio recording.

As a first step, we pre-filter some of the detected notes. Notes that are too quiet count as invalid. Furthermore, we remove any note that is a multiple of 12 half-steps higher than a note that is currently ringing (their octaves). In addition, to add robustness, we do not send octaves of the previously played note (even if the previous note is not ringing anymore). The reason is that, depending on the audio setup and how a string is held, a note that is one octave higher than the actually previously played note might be detected as a loud note at some point. Hence, in order to avoid accidental activations, we chose this filtering approach, meaning that there cannot be a sequence of two items that are one octave apart in our menu layout. All other notes count as *valid*, but not every valid note is forwarded to the frontend. We elaborate in the following.

As mentioned above, the incoming raw audio blocks have a fixed length (Δ_B = 80 ms in our reference implementation). This means that a note that rings throughout multiple audio windows creates a start note and an end note event in each individual audio window throughout its ringing duration, even if the learner has only hit a single note. At the same time, just truncating the prediction and discarding notes starting at the beginning and end of each audio window can lead to missed real notes, if they happen to start at any side of the audio window. Therefore, as depicted in Figure 5, we also incorporate past audio samples. More specifically, instead of only using the current audio block B_n, we concatenate the previous audio block B_{n-1} and run *Basic Pitch*'s prediction on the concatenated block $P_n = B_{n-1}B_n$. For instance, in Figure 5, a note starts at the end of B_0, so it is uncertain, whether or not it is detected by *Basic Pitch*.

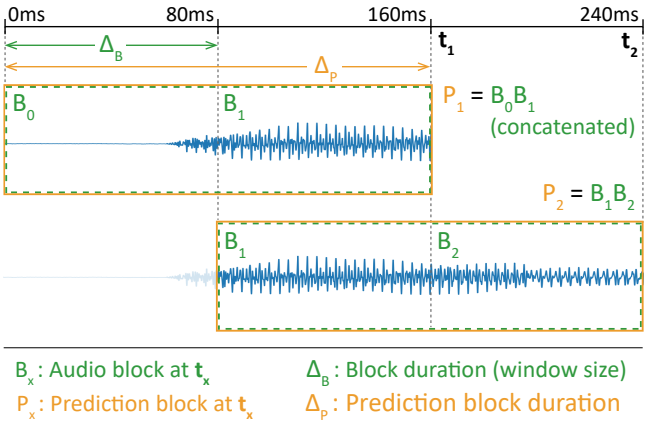

Figure 5: Illustration of two time steps in our usage of *Basic Pitch* in real-time. At each timestep t_n ($n \in \mathbb{N}$), we apply *Basic Pitch*'s MIDI conversion on P_n, which is the concatenation of the previous block of audio samples B_{n-1} and the current block B_n. Notes are only sent to the frontend, if the (global) time passed since the last valid note is longer than 100 ms.

However, the concatenation $P_1 = B_0B_1$ contains the full winding-up duration of the note, meaning that it will likely be detected. A valid note is only sent to the frontend if Δ_{min} =100 ms passed since the last valid note (i.e., ignoring invalid notes). For instance, if the note at the end of B_0 was already detected in a previous time step, then it is not sent again. This minimum time difference condition also ensures that a valid note that rings throughout multiple subsequent audio blocks is only sent once (because $\Delta_{min} > \Delta_B$).

6.2 Command and Bookmark Matching

As mentioned above, learners can create *song bookmarks* (see Section 4.1). We store those as wave files and run a *Basic Pitch* prediction on each of them when loading. While in the *Song Selection* state (`Selection UI` in Figure 4), we continuously run a *Basic Pitch* prediction to extract a *piano roll* (a 2D matrix that stores note states at every time step) on the last four seconds of audio input. We compare the live piano roll with the pre-recorded piano rolls of the song bookmarks. We additionally use *Dynamic Time Warping* on the piano rolls to account for slight differences in rhythm between the pre-recorded and the played snippet. If the overlap between two piano rolls is large enough, we open the *Tabs Player* with the song that the bookmark is from. The *commands* (see Section 4.2.2) are implemented in the same way as song bookmarks, i.e., when in the *Tabs Player* state, we match the live audio with the pre-recorded optional commands for unlocking and stopping playback.

7 Qualitative User Study

To gather feedback for both TabCtrl and GuitarPie, we conducted a qualitative study with five guitarists. We used the `Integrated` implementation (based on *Basic Pitch*) as described in Section 6.1.1 for the study. The study was approved by the ethics committee of our institution. In this section, we briefly describe the study and provide high-level insights, mostly focusing on the core interactions and findings. Additional study details, including more detailed participant feedback, can be found in the supplemental materials.

7.1 Participants

We recruited 5 participants (P1–P5) via convenience sampling. One of our workshop participants (P1) also participated in the design workshop (see Section 3). 3 participants identified as male, and 2 as female. They were between 28 and 35 years old. They self-rated their guitar skills as beginner (3), intermediate (1), and advanced (1), none were professional or expert. For the number of years of active guitar playing, they answered 1, 3, 10, 10, and 19 years. Tablature is used rarely (1), sometimes (2), and often (2). Sheet music is used never (1), rarely (1), sometimes (2), and often (1). They practice or jam with others never (1), often (3), and regularly (1). Common setups (multiple answers possible) were a screen on a desk (4), followed by printed on paper (3) and laptop/couch (2). One participant uses tabs on a phone, another one on a tablet.

7.2 Procedure

Before the study, we asked our participants to send us titles of songs they can already play or are currently practicing. We invited participants individually to our lab for about one hour each. First, we asked them to fill out a consent form and two survey forms

about demographics and prior experience with guitar playing and with tablatures. Afterward, the main part of the study commenced with the following procedure for each participant.

The participant was free to stand or sit in front of a 32" screen. We provided an electric guitar connected to an amplifier, which was connected to speakers as well as to our PC running TabCtrl. After a short introduction to TabCtrl and GuitarPie, we gave the participant about 10 minutes to try all features and get used to the menu. We gave hints and helped them when needed. Then, for the following 20–30 minutes, we asked them to perform various tasks that included all features (navigation, playback, song bookmarks, locking/unlocking, etc.). We asked participants to think aloud for the duration of the study. We recorded their voice and guitar sound as well as an over-the-shoulder video.

We concluded each participant's session with a semi-structured interview. Therein, we first asked for their general feedback and opinions as well as specific aspects such as whether and how they would change their setup if they would use TabCtrl during practice. As in our workshop (Section 3), we also asked the participant about their thoughts on other input modalities such as voice or hardware controllers. At the end of a session, the participant was paid 14 EUR and dismissed.

7.3 Results

7.3.1 GuitarPie Design. Overall, participants were able to learn using the menu and submenus throughout the study: "*by the end I was already more comfortable*" (P4). Our participants generally had no problems remembering what each icon stands for. Some were not intuitively clear, "*but using them once is enough*" (P1). P1 suggested to also show a legend in the menu and P3 to include a help button that toggles "*subtitles*" for menu items.

The six strings in our GuitarPie menu are in the same order as in a tablature. However, some participants, in particular ones who were less familiar with guitar tabs, indicated that they had a different mental model of the guitar string order: "*I feel like this was easier if it was like a mirror*" (P4). Other types of guitar interfaces, such as *Rocksmith* [49] and *guitARhero* [44] use such a mirrored representation. It remains to be tested, whether or not mirroring the menu would introduce inconsistencies in the case of tablature interfaces. Relatedly, P3 expected the up/down menu items in the instrument track menu to be flipped, while the other participants did not mention those items specifically.

As mentioned before, we use the item that opens a submenu as the new center similar to marking menus. P4 was confused about this and expected that submenus were at the same position as the main menu: "*I did not realize that [items] were changing position in submenus*".

7.3.2 Preferences and Customization. P3 found it conceptually weird to have appearance settings like the zoom level in the same submenu as playback options like increasing or decreasing speed. Others found this settings menu "*very logical*" because all options were about changing values and the "*plus and minus items are aligned*" (P5). Overall, the various preferences from different participants indicated that a menu customization option would be very useful when using GuitarPie for longer periods. This was also explicitly mentioned by some participants. For instance, P3 would only include the most needed items: "*I don't need the metronome because I already have the drums [...] I would just leave it aside and put my favorite stuff there*". P5 would "*definitely move [items] around [and] change the string you can open the menu with*".

7.3.3 Changing Practice Setups. We asked participants whether and how they would change their practice setup. The responses seem to depend on how they usually use tabs. Some use them for playing along, others only for looking up riffs (P3: "*when I look at the tab, I want to study it; I want to sit and study the tab*"). Similarly, P2 stated they "*would probably stay seated*". Others would use the ability to control tablature from further away to improve their setup: "*I can take a better position when I don't have to walk back to the PC*" (P1). "*I would have more freedom where to place my device. [...] When you live in a dorm you don't have much space. [...] I could put it on my desk and use it remotely*" (P5). For P1, this aspect differs between home practice and band practice, stating that marking is easier with a mouse when sitting right next to it, whereas in a band setting everything should be controllable with the guitar. P1 elaborates: "*Often when I [go to the large screen to change settings and] bow forward, I slam the guitar into someone*". This envisioned situational use is similar to the study conclusion of *Stretchy Strap*, which is also envisioned to be useful in specific circumstances and also based on individual preference [5, Sect. 4.1].

7.3.4 Emerging Behaviors. P1 tried on their own initiative and intuitively used palm muting while using the menu and was able to use the menu confidently with only few miss-inputs. Palm-muted notes are more distinct compared to playing legato (especially when playing fast) and unintended oscillations caused by accidentally touching strings are reduced. P3 used an alternative to locking the menu: Whenever they wanted to practice a part without the menu opening, instead of using the lock function, "*I can just turn down the [volume knob of the guitar] and practice*". In our setup this also disables amplified audio, but P3 considered the non-amplified guitar sound sufficient for quickly practicing a specific riff.

7.4 Revisiting Design Goals

We designed GuitarPie to enable audio-based control without having to look up or memorize note sequences (see Section 3.4 and Section 5). Even though some icons were not immediately obvious, participants were able to explore the functionalities, which is enabled by GuitarPie's spatial menu representation. This indicates that our *discoverability* criterion has been met. Furthermore, the participants became faster with frequently occurring sequences. Most prominently, the participants seemed to develop muscle memory for the sequence *Open Menu* → *Reset Playhead* → *Play*. Furthermore, users were able to record and use custom commands, e.g., to unlock the menu. However, the study additionally revealed that, besides custom commands, the menu layout and other aspects of GuitarPie should be customizable as well. We discuss this in the next sub section.

7.5 TabCtrl Refinements

Based on the participants' feedback and our observations, we added additional features and options to our GuitarPie instance in TabCtrl (also see Video Figure C in the supplemental materials).

Users can flip the order of the menu strings based on their preference and mental model of the fretboard representation (lowest pitch at lowest position like in the tabs *versus* seen as if mirroring the physical fretboard with the high E-string at the bottom).

Due to the different individual preferences in terms of menu layout (see Section 7.3.2), we made the menu customizable (Figure 6). When the menu is open, users can simply right-click on a location in the string-fret grid to open the list of operations, plus the option to remove the menu item, if there is one at that location. Choosing an operation moves the respective item from its previous location to the chosen slot, and overwrites the item that was there previously (if there was one). This way, users can move operations that they use often to the most convenient locations.

Furthermore, to avoid accidental activations especially while learning GuitarPie, we added a *double hit* option. If active, the same note needs to be played twice before the menu item gets activated. This way, learners can first assess whether they hit the intended item (as the item flashes, but without executing its operation) and then quickly play the same note again to confirm.

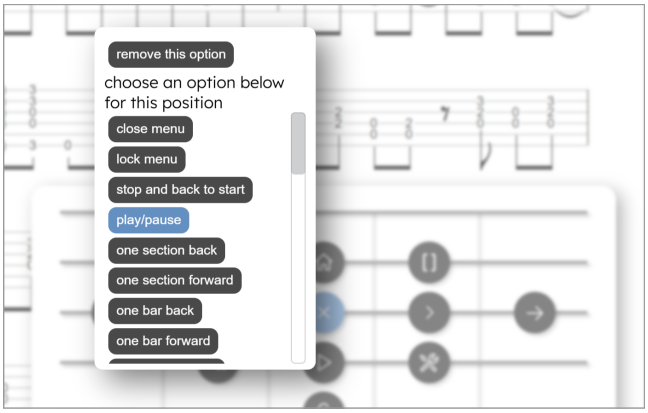

Figure 6: Menu layout customization. By right-clicking on a pie menu item or empty slot, a pop-up lets users choose the operation to be associated with this slot.

8 Discussion and Future Work

In this section, we position our work within the broader literature on digital guitar practice tools. We discuss how previous research can be combined with ours in the future to improve digital guitar practicing tools or for other use cases.

8.1 Combination with Sensing Hardware

Many previous approaches involve physical augmentations of the guitar [31, 37, 48, 51], including the use of magnetic [7, 38], capacitive [35], or pressure sensing [17, 50]. Hardware-based sensing is orthogonal to audio-based approaches, but could also be synergetic in some cases. For instance, a limitation of our audio-only approach is that it requires custom gestures for explicitly stopping playback and for unlocking the menu. Having the option to combine our audio-based approach with other sensing approaches has potential for integrated multimodal approaches such as applying pressure to the pick to toggle an audio-based menu or to confirm a selection.

8.2 Visually Augmenting the Guitar

Many previous guitar tutorial and practicing systems support visually co-located augmentations to enhance the learning experience. *Let's Frets* [35] and the similar *Fretlight*[8] physically augment the guitar with LEDs next to the frets for visual guidance. Besides physically modifying the guitar, researchers also investigated the use of augmented reality to enhance guitar tutorials and practice with co-located instructions [14, 23, 32, 39, 44]. We implemented our approach for tablature interfaces displayed on conventional screens. However, our approach can also be combined with unconventional output methods, which could be particularly interesting for interfaces, where mouse and keyboard are not even available. For instance, using the guitar's audio signals could also be used to control an AR/VR-based song practice system, where the virtual pie menu can then be co-located with the frets without requiring high-precision tracking. A minimal variant could use a conventional screen and a webcam to render the menu superimposed on a mirrored video feed of the fretboard around the learner's hand—thereby directly addressing a limitation found in our study, as this could help beginners hit the correct string and fret more easily.

8.3 Competing or Complementary Techniques

In this work, we focused exclusively on using guitar audio. Even though this modality can be used in tandem with other modalities, it is unknown in which situations one is better than the other, or in which situations they can be complementary. For instance, voice input can also be an alternative to conventional mouse and keyboard input while holding the guitar. Our qualitative evaluation solely focuses on how guitarists learn and use GuitarPie, specifically in a tablature control context. Hence, there are future research opportunities that could quantitatively compare those input methods (including cost for context switching, user preference, and more) with various different setups. To mitigate confounding factors like familiarity and novelty effects, this will require dedicated multi-day lab studies or deployments.

8.4 Other GuitarPie Use Cases

In this work, we presented the design of GuitarPie and applied it in a tablature control context. However, the technique can potentially also be used for other use cases that would normally involve switching between playing the guitar and interface control. For instance, navigating tutorial videos requires playback and navigation operations similar to those of tablature interfaces. Future use cases can also go beyond tabs and video navigation. For instance, audio recording software contains operations that could potentially be controlled from a distance via guitar audio, including starting and stopping the recording, as well as other operations such as effect adjustments or choosing a file folder when loading or saving. Similarly, remote jamming applications [25] often require to let go of the guitar to (un)mute, (de)activate the camera, and more. Another related opportunity is to use GuitarPie as a T9 text entry method for simple chat messages, e.g., during collaborative remote practice. Overall, exploring the applicability and ceiling of the GuitarPie technique in different contexts yields various future research opportunities at the intersection of music informatics and HCI.

[8] https://fretlight.com (accessed: 12th of July, 2025)

9 Conclusion

We presented our GuitarPie technique, which uses audio signals from an electric guitar as input and spatially represents menu items based on the grid-like layout of a guitar fretboard. GuitarPie's design was informed by a design workshop as well as by considering constraints and properties of guitar audio signals. Furthermore, we implemented a tablature interface called TabCtrl, which features a specific instance of GuitarPie as well as additional guitar audio input methods to make several functionalities of tablature configuration and playback controllable without letting go of the guitar. GuitarPie also has the potential to be used in use cases beyond tablature interfaces, such as effect control or remote jamming.

Acknowledgments

We thank the participants of our design workshop for their great ideas and inputs, as well as the participants of our qualitative study for their valuable feedback.

This work was supported by the *Alexander von Humboldt Foundation* (funded by the *German Federal Ministry of Education and Research*), the *Cyber Valley Research Fund*, and by the *German Research Foundation (DFG)* project *495135767*.

References

[1] Pedro Alonso, Raquel Cortina, Francisco J Rodríguez-Serrano, Pedro Vera-Candeas, María Alonso-González, and José Ranilla. 2017. Parallel online time warping for real-time audio-to-score alignment in multi-core systems. *The Journal of Supercomputing* 73, 1 (2017), 126–138.

[2] Andreas Arzt, Werner Goebl, and Gerhard Widmer. 2015. Flexible score following: The piano music companion and beyond. *Proceedings of the Third Vienna Talk on Music Acoustics* 16 (2015), 19.

[3] Andreas Arzt and Gerhard Widmer. 2010. Simple tempo models for real-time music tracking. In *Proceedings of the Sound and Music Computing Conference (SMC)*. Citeseer, 9.

[4] Andreas Arzt, Gerhard Widmer, and Simon Dixon. 2008. Automatic page turning for musicians via real-time machine listening. In *ECAI 2008*. IOS Press, 241–245.

[5] Juan Martinez Avila, Adrian Hazzard, Chris Greenhalgh, Steve Benford, and Andrew McPherson and. 2023. The Stretchy Strap: supporting encumbered interaction with guitars. *Journal of New Music Research* 52, 1 (2023), 19–40. https://doi.org/10.1080/09298215.2023.2274832 arXiv:https://doi.org/10.1080/09298215.2023.2274832

[6] Juan Pablo Martinez Avila, Adrian Hazzard, Chris Greenhalgh, and Steve Benford. 2019. Augmenting Guitars for Performance Preparation. In *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound* (Nottingham, United Kingdom) *(AM '19)*. Association for Computing Machinery, 69–75. https://doi.org/10.1145/3356590.3356602

[7] Adan L Benito Temprano and Andrew McPherson. 2021. A TMR Angle Sensor for Gesture Acquisition and Disambiguation on the Electric Guitar. In *Proceedings of the 16th International Audio Mostly Conference*. 256–263.

[8] Rachel M. Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert. 2022. A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 781–785. https://doi.org/10.1109/ICASSP43922.2022.9746549

[9] Marek Blok, Jan Banaś, and Mariusz Pietrolaj. 2021. IFE: NN-aided Instantaneous Pitch Estimation. In *2021 14th International Conference on Human System Interaction (HSI)*. 1–7. https://doi.org/10.1109/HSI52170.2021.9538713

[10] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. 1988. An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Washington, D.C., USA) *(CHI '88)*. Association for Computing Machinery, 95–100. https://doi.org/10.1145/57167.57182

[11] Supadaech Chanjaradwichai, Proadpran Punyabukkana, and Atiwong Suchato. 2010. Design and evaluation of a non-verbal voice-controlled cursor for point-and-click tasks. In *Proceedings of the 4th International Convention on Rehabilitation Engineering & Assistive Technology* (Shanghai, China) *(iCREATe '10)*. Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre, Article 48, 4 pages.

[12] Mads Græsbøll Christensen, Petre Stoica, Andreas Jakobsson, and Søren Holdt Jensen. 2008. Multi-pitch estimation. *Signal Processing* 88, 4 (2008), 972–983.

[13] Frank Cwitkowitz, Toni Hirvonen, and Anssi Klapuri. 2023. Fretnet: Continuous-Valued Pitch Contour Streaming For Polyphonic Guitar Tablature Transcription. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10094825

[14] Marta Sylvia Del Rio-Guerra, Jorge Martin-Gutierrez, Vicente A Lopez-Chao, Rodolfo Flores Parra, and Mario A Ramirez Sosa. 2019. AR Graphic representation of musical notes for self-learning on guitar. *Applied Sciences* 9, 21 (2019), 4527.

[15] Simon Dixon and Gerhard Widmer. 2005. MATCH: A Music Alignment Tool Chest. In *ISMIR*. 492–497.

[16] Matthias Dorfer, Jan Hajič jr, Andreas Arzt, Harald Frostel, and Gerhard Widmer. 2018. Learning Audio–Sheet Music Correspondences for Cross-Modal Retrieval and Piece Identification. (2018).

[17] Andreas Fender, Derek Alexander Witzig, Max Möbus, and Christian Holz. 2023. PressurePick: Muscle Tension Estimation for Guitar Players Using Unobtrusive Pressure Sensing. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, Article 80, 11 pages. https://doi.org/10.1145/3586183.3606742

[18] Raphael Foulon, Pierre Roy, and François Pachet. 2014. Automatic Classification of Guitar Playing Modes. In *Sound, Music, and Motion*, Mitsuko Aramaki, Olivier Derrien, Richard Kronland-Martinet, and Sølvi Ystad (Eds.). Springer International Publishing, 58–71.

[19] Markus Funk, Vanessa Tobisch, and Adam Emfield. 2020. Non-Verbal Auditory Input for Controlling Binary, Discrete, and Continuous Input in Automotive User Interfaces. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, 1–13. https://doi.org/10.1145/3313831.3376816

[20] Maarten Gasser, Martin Grachten, Andreas Arzt, and Gerhard Widmer. 2013. Automatic alignment of music performances with structural differences. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 607–612.

[21] Takahito Hamanaka, Daisuke Sakamoto, and Takeo Igarashi. 2014. Aibiki: supporting shamisen practice with adaptive automatic score scroll. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology* (Funchal, Portugal) *(ACE '14)*. Association for Computing Machinery, Article 13, 10 pages. https://doi.org/10.1145/2663806.2663839

[22] Florian Henkel and Gerhard Widmer. 2021. Multi-modal Conditional Bounding Box Regression for Music Score Following. In *2021 29th European Signal Processing Conference (EUSIPCO)*. 356–360. https://doi.org/10.23919/EUSIPCO54536.2021.9616287

[23] Frank Heyen and Michael Sedlmair. 2022. Augmented Reality Visualization for Musical Instrument Learning. In *Ismir 2022 Hybrid Conference*.

[24] Tung-Sheng Huang, Ping-Chung Yu, and Li Su. 2023. Note and Playing Technique Transcription of Electric Guitar Solos in Real-World Music Performance. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10095225

[25] Inc JamKazam. 2025. JamKazam. https://jamkazam.com/. (Accessed: 3rd of April 2025).

[26] Cyril Joder, Slim Essid, and Gaël Richard. 2013. Learning optimal features for polyphonic audio-to-score alignment. *IEEE Transactions on Audio, Speech, and Language Processing* 21, 10 (2013), 2118–2128.

[27] Rainer Kelz, Sebastian Böck, and Gerhard Widmer. 2019. Deep polyphonic ADSR piano note transcription. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 246–250.

[28] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. 2018. CREPE: A convolutional representation for pitch estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 161–165.

[29] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. 2018. CREPE Pitch Tracker. https://github.com/marl/crepe. (Accessed: 3rd of April 2025).

[30] Yuta Kusaka and Akira Maezawa. 2024. Mobile-AMT: Real-Time Polyphonic Piano Transcription for In-the-Wild Recordings. In *2024 32nd European Signal Processing Conference (EUSIPCO)*. 36–40. https://doi.org/10.23919/EUSIPCO63174.2024.10715008

[31] Livid. 2025. Guitar Wing. https://lividinstruments.com/products/guitar-wing/. (Accessed: 3rd of April 2025).

[32] Markus Löchtefeld, Sven Gehring, Ralf Jung, and Antonio Krüger. 2011. Using mobile projection to support guitar learning. In *Smart Graphics: 11th International Symposium, SG 2011, Bremen, Germany, July 18-20, 2011. Proceedings 11*. Springer, 103–114.

[33] Robert Macrae and Simon Dixon. 2010. A guitar tablature score follower. In *2010 IEEE International Conference on Multimedia and Expo*. 725–726. https://doi.org/10.1109/ICME.2010.5582963

[34] Karola Marky, Julian Fischer, Max Mühlhäuser, and Andrii Matviienko. 2021. Investigating Page Turning Methods for Sheet Music during Piano Play. In *Adjunct Publication of the 23rd International Conference on Mobile Human-Computer Interaction* (Toulouse & Virtual, France) *(MobileHCI '21)*. Association for Computing Machinery, Article 18, 6 pages. https://doi.org/10.1145/3447527.3474863

[35] Karola Marky, Andreas Weiß, Andrii Matviienko, Florian Brandherm, Sebastian Wolf, Martin Schmitz, Florian Krell, Florian Müller, Max Mühlhäuser, and Thomas Kosch. 2021. Let's Frets! Assisting Guitar Students During Practice via Capacitive Sensing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[36] Juan Pablo Martinez Avila. 2022. *Embodied interaction with guitars: instruments, embodied practices and ecologies*. Ph. D. Dissertation. University of Nottingham.

[37] Carolina Brum Medeiros and Marcelo M Wanderley. 2014. A comprehensive review of sensors and instrumentation methods in devices for musical expression. *Sensors* 14, 8 (2014), 13556–13591.

[38] Fabio Morreale, Andrea Guidi, Andrew McPherson, et al. 2019. Magpick: an augmented guitar pick for nuanced control. New Interfaces for Musical Expression.

[39] Yoichi Motokawa and Hideo Saito. 2006. Support system for guitar playing using augmented reality display. In *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 243–244.

[40] Meinard Müller. 2007. Dynamic time warping. *Information retrieval for music and motion* (2007), 69–84.

[41] Ondrej Polacek, Zdenek Mikovec, Adam J. Sporka, and Pavel Slavik. 2011. Humsher: a predictive keyboard operated by humming. In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility* (Dundee, Scotland, UK) *(ASSETS '11)*. Association for Computing Machinery, 75–82. https://doi.org/10.1145/2049536.2049552

[42] Xavier Riley, Drew Edwards, and Simon Dixon. 2024. High Resolution Guitar Transcription Via Domain Adaptation. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1051–1055. https://doi.org/10.1109/ICASSP48485.2024.10446182

[43] Mengyi Shan and TJ Tsai. 2021. Automatic Generation of Piano Score Following Videos. *Transactions of the International Society for Music Information Retrieval (TISMIR)* 4, 1 (2021), 29–42.

[44] Lucchas Ribeiro Skreinig, Denis Kalkofen, Ana Stanescu, Peter Mohr, Frank Heyen, Shohei Mori, Michael Sedlmair, Dieter Schmalstieg, and Alexander Plopski. 2023. guitARhero: Interactive Augmented Reality Guitar Tutorials. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 29, 11 (Nov 2023), 4676–4685. https://doi.org/10.1109/TVCG.2023.3320266

[45] Adam J. Sporka, Torsten Felzer, Sri H. Kurniawan, Ondřej Poláček, Paul Haiduk, and I. Scott MacKenzie. 2011. CHANTI: predictive text entry using non-verbal vocal input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, 2463–2472. https://doi.org/10.1145/1978942.1979302

[46] Ting-Wei Su, Yuan-Ping Chen, Li Su, and Yi-Hsuan Yang. 2019. TENT: Technique-embedded note tracking for real-world guitar solo recordings. *Transactions of the International Society for Music Information Retrieval (TISMIR)* 2, 1 (2019).

[47] Verena Thomas, Christian Fremerey, Meinard Müller, and Michael Clausen. 2012. Linking Sheet Music and Audio-Challenges and New Approaches. *Multimodal Music Processing* 3 (2012), 1–22.

[48] Luca Turchet and Mathieu Barthet. 2019. An ubiquitous smart guitar system for collaborative musical practice. *Journal of New Music Research* 48, 4 (2019), 352–365.

[49] Ubisoft. 2022. Rocksmith. https://www.ubisoft.com/en-us/game/rocksmith/plus. (Accessed: 3rd of April 2025).

[50] Roy Vanegas. 2007. The MIDI Pick: Trigger Serial Data, Samples, and MIDI from a Guitar Pick. In *Proceedings of the 7th International Conference on New Interfaces for Musical Expression* (New York, New York) *(NIME '07)*. Association for Computing Machinery, 330–332. https://doi.org/10.1145/1279740.1279812

[51] Tim Vets, Jonas Degrave, Luc Nijs, Federica Bressan, and Marc Leman. 2017. PLXTRM: Prediction-Led eXtended-guitar Tool for Real-time Music applications and live performance. *Journal of New Music Research* 46, 2 (2017), 187–200.

[52] Bryan Wang, Meng Yu Yang, and Tovi Grossman. 2021. Soloist: Generating Mixed-Initiative Tutorials from Existing Guitar Instructional Videos Through Audio Processing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

[53] Yulun Wu, Weixing Wei, Dichucheng Li, Mengbo Li, Yi Yu, Yongwei Gao, and Wei Li. 2024. Harmonic Frequency-Separable Transformer for Instrument-Agnostic Music Transcription. In *2024 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6. https://doi.org/10.1109/ICME57554.2024.10688217

[54] Daniel Zielasko, Neha Neha, Benjamin Weyers, and Torsten W. Kuhlen. 2017. A reliable non-verbal vocal input metaphor for clicking. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. 40–49. https://doi.org/10.1109/3DUI.2017.7893316

[55] Graeme Zinck and Daniel Vogel. 2022. Evaluating Singing for Computer Input Using Pitch, Interval, and Melody. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, Article 213, 15 pages. https://doi.org/10.1145/3491102.3517691

Appendix - Affinity Diagram of the Design Workshop

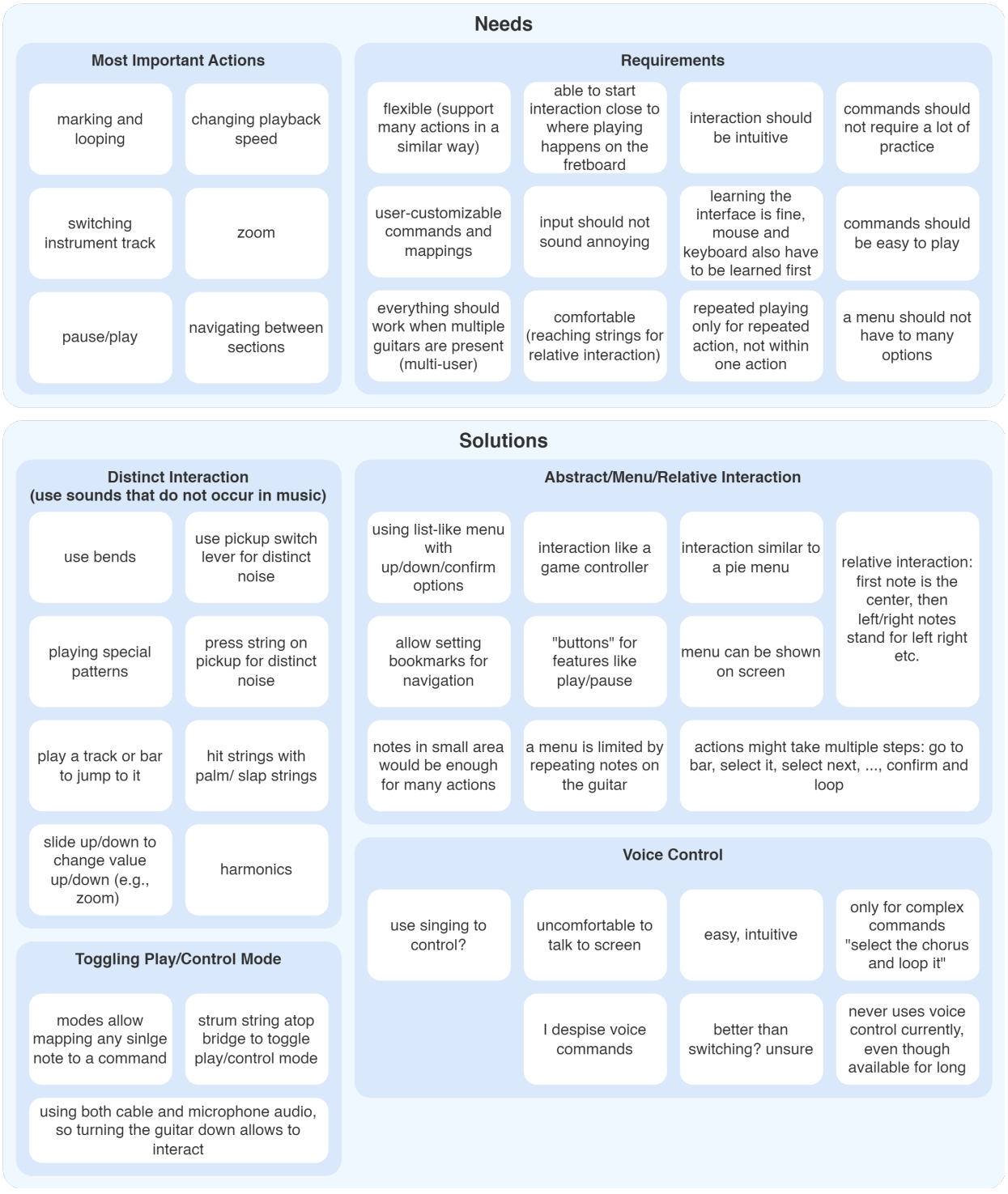

Figure 7: Affinity diagram with the outcomes of the design workshop. We abstracted ideas and clustered them by topic.