ARPiano Efficient Music Learning Using Augmented Reality

Fernando Trujano(✉), Mina Khan, and Pattie Maes

Massachusetts Institute of Technology, Media Laboratory, 77 Massachusetts Avenue, Cambridge 02139, USA
Trujano.Fernando@gmail.com

Abstract. ARPiano uses a MIDI keyboard and a multifunction knob to create a novel mixed reality experience that supports visual music learning, music visualizations and music understanding. At its core, ARPiano provides a framework for extending a physical piano using augmented reality. ARPiano is able to precisely locate a physical keyboard in order to overlay various objects around the keyboard and on individual keys. These augmented objects are then used for music learning, visualization and understanding. Furthermore, ARPiano demonstrates a novel way to utilize the keys in a piano as an interface to interact with various augmented objects.

Keywords: Augmented reality · Piano · Learning · Education · Music

1 Introduction

This paper describes ARPiano, a mixed reality application that augments a physical keyboard. ARPiano uses a MIDI (Musical Instrument Digital Interface) keyboard and a multifunction knob to create a novel mixed reality experience that supports visual music learning, music visualizations and music understanding. ARPiano is able to precisely locate a physical keyboard in order to overlay various objects around the keyboard and on individual keys. These augmented objects are then used for music learning, visualization and understanding. Furthermore, ARPiano demonstrates a novel way to utilize the keys in a piano as an interface to interact with various augmented objects.

1.1 Goals

ARPiano is designed to allow for better visual music learning, music visualization and music understanding.

Music Learning. The main goal of ARPiano is to facilitate visual music learning using augmented reality. Traditional music learning focuses on reading symbols in the form of sheet music. However, these symbols are abstract and there is no intuitive correlation between the symbol and the note that it represents. For example, notes of different lengths are differentiated by whether the symbols are filled in or not (quarter note and half notes)

T.-T. Wu et al. (Eds.): ICITL 2018, LNCS 11003, pp. 3–17, 2018.
https://doi.org/10.1007/978-3-319-99737-7_1

or whether they have an extra tick attached to them (eighth notes and sixteenth notes). This notation is not immediately intuitive and requires that the user learn the symbols in order to make sense of them. Current theories of learning show that using multiple modalities in learning is an effective strategy to form mental constructs and schemas [1–4]. The visual aspect of traditional music learning depends on these abstract symbols and could be stronger.

ARPiano aims to facilitate visual music learning by providing a deeper connection between the song the user is learning and what they are seeing. This is accomplished by intuitive and spatially accurate visuals, real time feedback, and performance statistics. ARPiano visually renders a song in the form of a sequence of cuboids where the length of the rectangle represents the length of the note and the position relative to the keyboard represents the note value.

Music Visualization and Understanding. ARPiano also allows for music visualization and music understanding. Music visualization is accomplished by emitting a sprite from the physical key location whenever the key is pressed. This provides the user with a visual counterpart to what they are hearing and allows them to better see patterns in the music. Additionally, while the user is playing, ARPiano is able to augment the interface to point out specific musical artifacts, such as chords, which aids in music understanding.

2 Previous Work

Music learning software is not a new field and there are many applications available that try to facilitate music learning by creating visuals or providing performance statistics. However, none of the applications provide spatially accurate visuals.

2.1 Smart Music

Smart Music [5] is a music learning software that provides helpful feedback on a player's performance. A user is displayed the sheet music as well as a bar indicating the current measure that they are on. While playing a song, Smart Music uses a microphone to determine whether the user is playing the correct notes. While Smart Music provides helpful statistics to the user, such as which sections had the most mistakes, it still relies on sheet music and does not offer a more intuitive musical representation of a song.

2.2 Synthesia

Synthesia [6] is a piano learning software that, like ARPiano, renders songs in the form of a piano roll. Each note in the song is represented by a rectangle that slowly approaches its corresponding key on a virtual keyboard on the display. This representation is more intuitive than sheet music and allows the user to see patterns in the song. Synthesia is also able to track a player's performance and provide statistics to help them guide their practice. While Synthesia provides an intuitive visual representation of the song, it does

so on a traditional 2D display and requires that the user look back and forth between the virtual keyboard on the display and the physical keyboard they are playing on.

2.3 Andante/Perpetual Canon

Andante [7] and Perpetual Canon [8] are part of a collection of research projects developed by Xiao Xiao from the MIT Media Lab. They augment a piano with a projector and are able to provide visualizations of pre-recorded or live music. These projects did not however, focus on music learning or analyzing what is being played.

3 Design

ARPiano is implemented as a Unity application that runs on a Hololens and wirelessly receives messages from a server running on a PC. The application augments the physical keyboard with sprites and annotations and receives input from the keyboard in order to support various interactions. ARPiano adopts a modular design in the form of Keyboard Components (see Sect. 4).

3.1 Integrated Sensors

ARPiano uses a MIDI Keyboard and a multifunction knob for input. Both of these devices communicate over USB. Since the Hololens does not have any USB ports, additional steps had to be taken to integrate these devices into a Mixed Reality experience. The application also uses the built in Hololens camera to get the initial position of the keyboard in three dimensional space.

The keyboard sends MIDI commands to the computer over USB and is the main interface for ARPiano. When a key is pressed the keyboard emits the corresponding sound and sends a MIDI message to the connected computer. MIDI is well established and documented so many libraries exist to parse these messages. I used the open source Midi Jack Unity plugin [9] to parse each incoming MIDI message into its corresponding channel, note and velocity. The computer then sends each event (either noteOn or noteOff) to the Hololens using the HoloToolkit sharing service. These messages are then used by the Hololens to support various interactions.

The multifunction knob serves as an additional controller to change several parameters of the keyboard. An augmented interface can be rendered on top of the controller to show which parameter is being changed (see Sect. 4.3). I used the Griffin Power mate to map knob rotations and presses to different keypress events. These events are then picked up by the computer using Unity's Input library and sent to the Hololens via the Sharing Service. The Hololens then decides what to do depending on the current state of the keyboard.

Lastly, ARPiano uses the built in Hololens camera to detect the three dimensional position of the first and last keyboard keys as described in the following subsection.

3.2 Physical and Augmented Keyboard

At its core, ARPiano consists of a main Keyboard class that handles incoming messages from the server and is responsible for the initial calibration. The keyboard class is responsible for parsing these messages and making them available to Keyboard Components (see Sect. 4). Each message consists of the pressed note (as a midi integer) and the velocity of the pressed note (as a float).

Calibration of the physical keyboard allows the Hololens to know precisely where each key on the keyboard is in three dimensional space. This process is done by placing two key-sized Vuforia markers on the first and last key of the keyboard. When the application is first opened, the user is instructed to look at the first and last key of the keyboard until the keys turn green, indicating successful detection of the markers (Fig. 1). In order to accommodate keyboards with different number of keys, the user then presses the first and last key. This finalizes the calibration procedure and allows the Keyboard class to accurately determine the correct position of the keyboard and each individual key therein.

Fig. 1. Calibration ensures that the Hololens knows the exact location of the physical keyboard. The markers show the perceived location of the first and last keys. (Color figure online)

The Keyboard Class exposes several methods such as GetNotePosition which returns the position of a key in the physical space given the corresponding midi number. This allows components like NoteSpawner to place sprites in the position of a corresponding key (see Sect. 4.4). The Keyboard class also allows KeyboardComponents to subscribe to three different events: calibrationComplete, noteOn, and noteOff.

3.3 Component Menu

The Component Menu utilizes the keyboard keys as an interface to enable or disable individual Keyboard Components. To open the menu, the user must press and hold the first key in the keyboard. While the key is being held, a loading ring appears on top to indicate how much longer until the menu opens. ARPiano avoids accidental opening of the menu by requiring the user to hold the lowest key for a couple of seconds, which does not happen often in regular music.

Once the menu is opened, all keyboard components are paused and icons representing keyboard components are rendered on the keyboard. Each icon consists of a colored rectangle that spans four white keys on the keyboard along with a three dimensional representation of the keyboard component, as seen in Fig. 2. The rectangle appears

green or white depending on whether the corresponding component is enabled or disabled. To change the state of a component, the user simply needs to press any of the keys under the icon. Once the state of a component changes, the menu closes. Alternatively, the user can press the first key again to close the menu.

Fig. 2. 3D icons representing Keyboard Components are overlayed on top of the keyboard. From left to right, the icons represent: Keyboard Labels, Note Spawner, Chord Suggestion, Chord Detector and Song Tutorial. In this figure, the Note Spawner component is enabled. (Color figure online)

4 Keyboard Components

ARPiano is designed to allow for easy addition of new features in the form of Keyboard Components. A Keyboard Component subscribes to messages from the main Keyboard class and can be enabled or disabled independently of other Keyboard Components. Keyboard Components use helper methods from the Keyboard class to correctly position objects relative to the keyboard or the desired key.

At this time, ARPiano consists of six different Keyboard Components that enable music learning, understanding and visualization. The main goal of ARPiano is to facilitate visual music learning through augmentations on the physical keyboard. This is accomplished by the Chord Suggestion, Keyboard Labels and Song Tutorial Components. Music visualization complements the music listening experience by providing a visual counterpart to the song being played. It can also provide a history to each note and allow for easier recognition of patterns in a song. In ARPiano the Note Spawner and Flashy Visualizer Components visualize the notes as they are being played. Music understanding reinforces the music learning aspect by creating deeper connections between what is being played and what is being learned. ARPiano allows users to better understand music by pointing out interesting note intervals through different colors and annotations. The Note Spawner and Chord Detector components aid in music understanding.

4.1 Chord Suggestion

The Chord Suggestion component gives hints to a user as they are learning different chords. When a key is pressed, other keys next to it are highlighted to indicate that those keys, together with the pressed key, form part of a chord (Fig. 3). More specifically, the pressed key is treated as the root key in the cord and the third and fifth notes of the triad

are highlighted. Different color highlights indicate different chord qualities, though at this time only major and minor chords are supported.

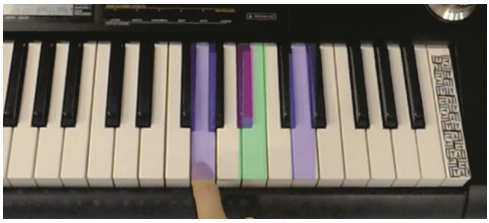

Fig. 3. The Chord Suggestion component highlights keys that can form chords. In this figure, both major (green highlight) and minor (pink highlight) chords are suggested with shared keys being highlighted blue. (Color figure online)

The Chord Suggestion component works by doing MIDI math on the pressed midi notes. Since the intervals between the notes in the triads are fixed regardless of the root note, this is a straightforward calculation.

4.2 Keyboard Labels

The Keyboard Labels component adds note labels to each key of the keyboard to assist the user in recognizing the position of notes in the keyboard (Fig. 4). The Component loops from the first key of the keyboard (as determined during the calibration procedure) to the last key and adds the corresponding note label. In MIDI, each note is represented as an integer with Middle-C being 60 and subsequent indices corresponding to the next semitones (so 61 corresponds to C#, for example). Knowing this, it is possible to label all of the keys in the keyboard.

Fig. 4. The Keyboard Labels component augments each key by labeling its corresponding note.

4.3 Song Tutorial

The Song Tutorial Component is the most complex Keyboard Component in ARPiano. It enables music learning by allowing the user to play any MIDI song in a visual and intuitive way. The component works by rendering a virtual piano roll of the song that precisely matches the physical piano. That way, as the piano roll moves down, the user can press the indicated keys in a similar way to popular music games like Guitar Hero. The speed and current position of the song can be adjusted using the multifunction knob and interactive performance statistics are displayed at the end in order to increase practice efficiency.

PianoRoll. The piano roll consists of a series of TutorialNotes depicted as cuboids whose position in the left/right axis represent the key to be played while the position in the vertical axis represents the time when the key should be played. Furthermore, the length of the cuboid represents the amount of time the note should be held for, and black keys are differentiated from white keys by the their position and color (Fig. 5). The piano roll starts on top of the keyboard and slowly moves down, causing some Tutorial Notes to lay directly on top of the keys of the keyboard. Each time a Tutorial Note does this, the user is instructed to press the key on the keyboard. By following these simple instructions, a melody can be played.

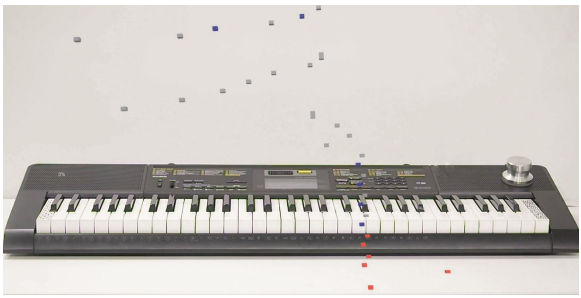

Fig. 5. SpectatorView image of AR PianoRoll representation of Beethoven's Für Elise

The piano roll is rendered by taking any MIDI song and translating it to a Tone.js compatible JSON file. Tone.js is a java script library to parse and create MIDI files, and includes an automatic script to convert a midi file to a json file [10]. It was chosen specifically to avoid working with MIDI files to speed up development as I was already familiar with parsing JSON files in Unity. From the MIDI file, the JSON file is able to obtain a list of notes, each consisting of a time, duration, and midi integer. These can be easily mapped into their corresponding Tutorial Note cuboids using the main Keyboard class.

As the piano roll moves down, the user must press the correct keys in order to play the melody. In addition to the auditory feedback the user gets from the piano, Tutorial Notes change color to indicate whether they were correctly played or not. Note verification, as described above, is accomplished by maintaining a list of notes are currently touching the keyboard and thus, should be played. If a note in this list is played, it is marked as correct and turns green, otherwise if the note leaves the keyboard without being played, it is marked as incorrect and turns red. This visual feedback provides another way for the user to gauge their performance in real time as they are playing the song.

Song Adjustments. The Song Tutorial Component allows the user to adjust the current position of the piano roll and the speed at which it moves down by turning a multifunction knob. The song can also be paused and resumed by pressing the knob. More specifically, I connected a Griffin Power mate to the server computer and mapped various types of rotations and presses to different keyboard presses that get forwarded to the Hololens

through the sharing service. The knob supports two kinds of rotations: a regular rotation, and a pressed rotation. These are both sent to the Hololens.

The knob is augmented with two sets of half rings around it that represent two song parameters. The outer ring depicts the current position of the song and the inner ring represents the speed at which it is being played. When the knob is turned, the corresponding ring value changes and its exact value is shown on top of the ring (Fig. 6). The knob UI is an interesting demonstration of how augmented reality can create a multifunction interface using a single device. The knob is kept on the keyboard close by so that it can be reached easily. During music practice, a user is able to use the visual feedback and quickly "scroll" back to a difficult section in order to practice it at a slower tempo, for example.

Fig. 6. The knob can be used to see and control the position of the song being played (left) or the speed at which it is being played (right)

Song Tutorial Results. After a song is played, the Song Tutorial component displays helpful statistics about the performance in the form of Song Tutorial Results. These results use the keyboard as an interactive canvas to visually show different statistics. An arrow is rendered on the last key to signify that pressing it will render the next available result. Currently, ARPiano supports two types of results: mistakes over time, and mistakes per key.

The first graph maps each key in the keyboard to a section of the song and color codes the keys depending on the number of mistakes that occurred during that section. This allows the user to quickly see which sections of the song need more practicing, as seen in Fig. 7. Additionally, pressing a key will automatically scroll to that section of the song so the user can practice it immediately.

The second graph color codes each key to the ratio of times that that key was pressed correctly. This allows the user to quickly identify which hand needs more practice. Pressing a key will reveal the exact number of times that that key was supposed to be played, and the number of times it was actually played.

Song Tutorial Results demonstrate a novel way to utilize the keyboard as an interactive canvas while providing useful statistics to help maximize practice efficiency.

Fig. 7. At the end of a Song Tutorial, the keyboard is transformed into a visual representation of the user's performance. In this graph, it is clear that the user made more mistakes in a specific section near the beginning of the song.

4.4 Note Spawner and Flashy Visualizer

The Note Spawner and Flashy Visualizer both emit sprites when a key is pressed from the location of the pressed key. These components subscribe to the noteOn and noteOff events and use the Keyboard's helper methods to position the sprites at the correct location.

The Flashy Visualizer, as the name implies, aims to complement a live music performance by providing colorful visuals corresponding to the notes being played. This Component renders various sprites when a key is pressed, or when a combination of keys is pressed (such as a chord) in order to provide a visual counterpart to the notes being played.

The Note Spawner emits a cuboid from the pressed key that slowly travels up. The length of the emitted sprite represents the length that the note was held for while the speed at which the sprite moves up represents the velocity of the corresponding key press. The NoteSpawner creates a temporal dimension to the music being played while allowing the user to see patterns within the song (Fig. 8).

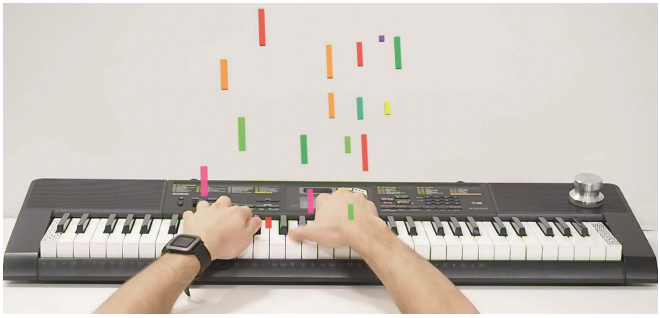

Fig. 8. The Note Spawner component emits cuboids as the user plays the keyboard in order to visualize rhythmic and tonal patterns in the song.

The Note Spawner Component can be configured to spawn notes whose color corresponds to that of a color wheel overlayed on top of the circle of fifths. This harmonic

coloring provides an interesting visualization of the note intervals so a close relationship in sound is described visually by a close relationship in color [11]. Using these colors, tonal patterns can be seen in many songs, such as Chopin's Prelude, Opus 28, No. 3 in G major, which remains entirely in a single tonal area [4]. The note positions and lengths in combination with the harmonic coding provide a way to observe the tonal and rhythmical patterns of a song in a way that sheet music can simply not capture.

4.5 Chord Detector

The Chord detector component makes the user more aware of chords by creating an annotation of the chord name every time a chord is played (Fig. 9). By reminding the user of the chord name whenever a chord is played a deeper connection can be formed between the notes and the corresponding chord. I hypothesize that this aids in music understanding by making the user more aware of common chord intervals. Similarly to the Chord Suggestor, the Chord Detector works by analyzing the intervals of all notes being pressed. On each key press, the Chord Detector analyzes every subset of three notes and tries to match them to a major or minor chord. Once the chord is detected, an annotation with the chord name is rendered on top of the root note. The positioning of the annotation allows the user to see which inversion of the chord was played.

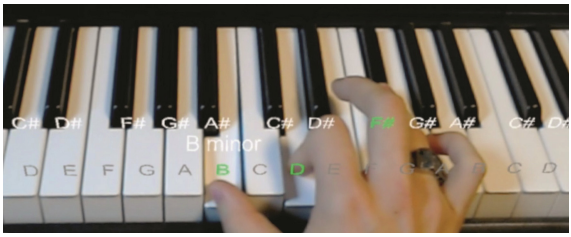

Fig. 9. The Chord Detector component analyzes the keys as they are pressed in order to label chords when they are played. This figure also shows how multiple components (ChordDetector and KeyboardLabels) can be enabled at once.

5 Evaluation

I designed a preliminary experiment to evaluate the efficiency of the visual music learning aspect of ARPiano. More specifically, I wanted to analyze the strengths and weaknesses of ARPiano by asking non-piano players to try to play various melodies without practicing them ahead of time. As a baseline comparison, I used the Synthesia software described in Sect. 2.2.

Each participant was asked to play 4 songs in total, 2 using ARPiano, and 2 using Synthesia. I chose the following songs: Twinkle twinkle (easy difficulty, both parts), Beethoven's Ode to joy (easy difficulty, both parts), Beethoven's Für Elise (medium difficulty, melody only), and Bach's Minuet in G (medium difficulty, melody only). The songs were chosen because of their popularity and varying degrees of difficulty.

For each participant, the starting method (ARPiano or Synthesia) and song pairing (one from each difficulty level) was randomly selected such that each song got played an equal amount of times on each method.

Users were briefly instructed on how to use each method and played a short chromatic exercise using the method in order to make sure they understood how to use the method correctly. After the method tutorial, each user played two songs using the selected method, filling out a quick survey in between songs.

After the first method was completed, the same exercise was conducted with the other method and the other two songs.

5.1 Study Results

I conducted this study with 14 participants (11 Female, 2 Male, 1 Agender) aged between 18–22. 57% participants had played instruments before (35.7% had played one instrument, 21.4% had played two), and exactly 50% of participants could read sheet music. None of the participants reported having played the piano before.

All participants agreed (7.1%) or strongly agreed (92.9%) that they "found ARPiano fun to use". In comparison, 35.5% of participants agreed (28.6%) or strongly agreed (7.1%) that they "found Synthesia fun to use". 35.5% of participants agreed that they would use Synthesia again to learn piano while 85.7% of participants agreed (21.4%) or strongly agreed (64.3%) that they would use ARPiano again. 85.7% of participants agreed (35.7%) or strongly agreed (50%) that "ARPiano was intuitive to use" while only 42.8% of participants agreed (21.4%) or strongly agreed (21.4%) that "Synthesia was intuitive to use".

After each performance, participants were ask to rate their performance based on their initial expectations. 78.57% of ARPiano players reported performing about the same (42.85%) or better than (25%) or much better than (10.72%) what they originally expected. In comparison, only 21.4% of Synthesia players reported performing about the same (14.3%) or better than (7.1%) what they originally expected, with 46.4% of them performing "much worse than" what they expected.

On average, users performed better using ARPiano and commented on the convenience of having the note sprites land on the corresponding physical key instead of on a virtual keyboard on the monitor. One participant was even able to play both songs (Ode to Joy and Bach Minuet) with fewer than 2 errors in each. Interestingly, for song sections with consecutive notes (key-wise), participants were often able to play the melodies correctly with both methods. However, ARPiano performed much better in sections where the notes were not consecutive. With Synthesia, users often missed any jumps in the notes because they were unable to determine how far to move their hand to hit the correct note and there was not enough time to look back and forth between the keyboard and the monitor. With ARPiano, users were able to see exactly where their hand should move to without having to look away from the keyboard.

It should be noted that the limited field of view of the Hololens made it difficult to play notes that were more than an octave an a half away from each other. For example, there is a section of Für Elise where a single high note must be played. All ARPiano players missed the jump since they could not see the note coming into the keyboard.

Synthesia players saw the jump, but usually hit the wrong note. These types of mistakes should decrease as AR technology improves.

6 Challenges

While ARPiano was designed to overcome many Hololens specific challenges, there were still different challenges to overcome in order to make ARPiano a reality. These include UI specific challenges when working with AR, imperfect Vuforia tracking and a lack of a music theory background.

For ARPiano all UI Elements must be relative to the keyboard. This means that to move something to the right it is not sufficient to move it in it's x direction as the definition of "right" depends on the initial position and rotation of the keyboard. I solved this by exposing the position, rotation, and directional vectors (forward, up, right) of the physical keyboard to other classes, and making all UI elements relative to that. However, this was not always sufficient as the Vuforia markers provided imperfect tracking.

For the most part, the Vuforia marker positioning was accurate to a centimeter which was good enough for the purposes of positioning objects relative to the keyboard. The rotation of the markers however, was not as accurate and caused issues with some UI elements, such as the piano roll in the Song Tutorial component. To support tilting of the piano roll, I initially rendered the piano roll vertically and rotated it around the right vector of the keyboard. I then moved the entire piano roll in a vector such that the first note would hit it's correct position. In theory, this should make all notes hit the correct position. However, due to small imperfections on the axis of rotation (i.e., the rotation of the keyboard as determined by the Vuforia markers) a small drift was introduced and the Tutorial Notes quickly became out of place, making it impossible to continue playing the song. I solved this challenge by expanding the Tutorial Note Class and making each note move towards its end location individually of all other notes. This effectively cancels out the imperfections on the rotation and guaranteed that all notes would always hit the correct spot on the keyboard.

This project required a surprising amount of knowledge of music theory. I had to familiarize myself with the underlying music theory to be able to implement components such as the Chord Detector. Some helper functions, particularly those in the keyboard class pertaining to the structure of notes in the keyboard, are written naively and inefficiently. While these do not have an adverse effect on performance, I'm sure I could find a more elegant way to write them if I had a stronger music theory background.

7 Future Work

ARPiano explores the different applications and benefits that an augmented instrument can provide. These applications range from music learning to music visualization and understanding. As it stands, with its various Keyboard Components, ARPiano is able to show the potential of an augmented keyboard to the above applications. However, the possibilities of an augmented instrument are far greater than those implemented in the current version of ARPiano.

Keyboards are not the only type of instruments that can benefit from augmentation. I hypothesize that many other instruments, such as string or brass instruments can benefit in their own way from various augmentations. These instruments would be trickier to implement because unlike a keyboard, they can move around while they are being played, which would require constant, real-time tracking from the headset. On the music visualization end, augmented instruments have direct applications to live performances where each instrument can be visualized independently in order for the audience to get a better feel and understanding on what each instrument is contributing to the overall song. For easier adoption, a smartphone version using current AR technologies like ARCore and ARKit can be created.

More research is needed to better understand the benefits of an augmented instrument in regards to music visual learning. For example, it would be interesting to see the a similar study as the one described in Sect. 5 over a longer time period to see whether the augmented keyboard allowed users to master a song more efficiently than other methods.

With an augmented instrument, music learning can take a more traditional approach or a completely different, gamified approach. With a traditional approach, the system could present the user with each song section and allow them to practice it independently. When the user has shown mastery of the current section, the next section can be rendered and practiced. Using devices such as the Muse [12] the system could be able to detect when the user is ready to move onto the next section by measuring the level of concentration of the user as they play. High brain activity would signify that the user is still struggling with the piece while low activity would signify that the section is likely committed to muscle memory since the user is more comfortable playing it. This allows the system to take full control of the learning space in an attempt to maximize the efficiency of the learning process.

An augmented keyboard also allows the possibility to take a completely different approach to music learning. For example, the learning experience could be masked in the form of various games where the correct combination of keys need to be pressed in order to pass each level. This experience would be especially appealing to kids, who may find it difficult to get started with a piano due to the lack of visual stimulation. Without even being aware of it, kids could be learning about chords structures while playing a fun and entertaining game.

8 Conclusion

ARPiano uses the Microsoft Hololens to augment a physical MIDI keyboard in order to facilitate visual music learning, understanding and visualization. This novel application is accomplished by combining existing peripherals such as a MIDI keyboard and multi-function knob with Augmented Reality technology.

ARPiano provides a way to track the position of each key on the keyboard in order to render various sprites and annotations on the physical keyboard depending on what the user is playing. With the Component Menu and Song Tutorial Results, ARPiano also demonstrates a way to use an existing physical peripheral as an intuitive interface to

various supported interactions. Lastly, ARPiano provides a helpful way to visualize music where both tonal and rhythmic patterns can be observed.

Before augmented reality, adding visual learning functionality to instruments required expensive hardware and visuals were limited to that hardware. With augmented reality, there is no limit to the type and position of objects that can be rendered on the keyboard. Furthermore, adding new functionality does not require any additional hardware as all new functionality can be virtually rendered into the user's field of view. I believe this opens the possibility to rethink the way we approach music learning.

Acknowledgments. Redacted to maintain submission anonimity.

References

1. Campbell, P.S., Scott-Kassner, C., Kassner, K.: Music in childhood: from preschool through the elementary grades. Schirmer Cengage Learning, Boston (2014)
2. Gault, B.: Music learning through all the channels: combining aural, visual, and kinesthetic strategies to develop musical understanding. Gen. Music Today **19**(1), 7–9 (2005). https://doi.org/10.1177/10483713050190010103
3. Morris, C.: Making sense of education: sensory ethnography and visual impairment. Ethnography Educ. **12**(1), 1–16 (2016). https://doi.org/10.1080/17457823.2015.1130639
4. Harmonic Coloring: A method for indicating pitch class (n.d.). http://www.musanim.com/mam/circle.html. Accessed 11 Dec 2017
5. Music Learning Software for Educators & Students (n.d.). https://www.smartmusic.com/. Accessed 11 Dec 2017
6. Synthesia, Piano for Everyone (n.d.). https://www.synthesiagame.com/. Accessed 11 Dec 2017
7. Xiao, X.: Andante: a walking tempo (n.d.). http://portfolio.xiaosquared.com/Andante. Accessed 11 Dec 2017
8. Xiao, X.: Perpetual Canon (n.d.). http://portfolio.xiaosquared.com/Perpetual-Canon. Accessed 11 Dec 2017
9. Takahashi, K.: MidiJack, 5 November 2016. https://github.com/keijiro/MidiJack. Accessed 11 Dec 2017
10. Tone.js (n.d.). https://tonejs.github.io/. Accessed 11 Dec 2017
11. Color Music Theory (n.d.). https://www.facebook.com/Virtuosoism/. Accessed 11 Dec 2017
12. How does Muse work? (n.d.). http://www.choosemuse.com/how-does-muse-work/. Accessed 11 Dec 2017
13. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**, 195–197 (1981)
14. May, P., Ehrlich, H.C., Steinke, T.: ZIB structure prediction pipeline: composing a complex biological workflow through web services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
15. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
16. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid information services for distributed resource sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)

17. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the grid: an open grid services architecture for distributed systems integration. Technical report, Global Grid Forum (2002)
18. National Center for Biotechnology Information. http://www.ncbi.nlm.nih.gov