Proceedings of RoCHI 2021

An Augmented Reality Piano Learning Tool

Andra Simion, Adrian Iftene, Daniela Gîfu

Faculty of Computer Science, "Alexandru Ioan Cuza" University of Iași, Romania {andra.simion, adiftene, daniela.gifu}@info.uaic.ro

ABSTRACT

Music learning (here, thinking of playing piano) without professor can be a real challenge. This can be difficult to achieve, given some traditional approaches to manage this process are still insufficiently satisfactory. Beyond the wellknown computer-assisted music learning tools (1) screenbased and (2) hardware-based, a promising solution for piano learning is based on augmented reality (AR). This technology provides an ideal environment for the student, but of course, it is not enough to become a career musician. In this paper, we propose an application that helps all those who want to learn to play the piano in a friendly way, starting with understanding sheet music. By projecting information onto the instrument, our AR system can create more direct interactions for the user.

Author Keywords

ELearning, Augmented reality, Playing piano.

ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces. H.3.2. Information Storage and Retrieval: Information Storage.

General Terms

Human Factors; Design.

DOI: 10.37789/rochi.2021.1.1.21

INTRODUCTION

Music learning represents a field that acquired importance to people for their health and well-being [1] due to the recent validation that attending music lessons can produce significant increases in psychological and cognitive areas [2]. However, learning a new instrument can be challenging as you cannot progress unless you know how to read sheet music [3]. Moreover, these days due the lack of time and financial constraints, people can't afford too many lessons assisted by a professor, developing wrong habits in practice [4]. There are some well-known computer-assisted music learning approaches [5], namely: (1) screen-based (e.g., Piano [6] or Yousician [7]), that require users to visually transfer information from the device to apply on the specific instrument or (2) hardware-based (e.g., The ONE Smart Piano [8]), that provides direct guidance to the detriment of the user's engagement - the problem to facilitate this process remains. Beyond these methods, a promising solution for computer-assisted piano learning is based on augmented reality (AR). AR is an interactive experience that can create a connection between the student and the system by defining more practical possibilities with the instrument. Azuma

gathered existing information from several sources to conceive a survey of AR [9], considered a starting point for any AR system. The use of AR in education has become something natural lately, with applications to learn geography [10], chemistry [11], computer science [12], or how to work in a team [13]. One of the most well-known computer-assisted piano systems is called Synthesia where you can add MIDI files for certain songs you want to listen to or even practice playing them. The software processes the files and formats the notes visually so that it always displays the keys you should press along with its duration. Unfortunately, using Synthesia has its own disadvantages, namely, you need to have a computer on which the program runs, and this computer needs to be placed somewhere behind the keyboard so that you can follow the notes.

The legitimate research question of this paper intends to answer: *How efficient is AR-based way of learning the piano?* This research intends to draw the basic knowledge level over which it intends to highlight the role of the ARassisted music education. We aim to develop a rapid, friendly, and efficient system, assisted by AR, to help those who dream of playing the piano.

The rest of paper is organized as follows: Section 2 presents a short overview of piano learning systems very useful in the musical practice of beginners, while Section 3 describes the prototype based on AR. Section 4 briefly discusses the evaluation of this new system before drawing some conclusions in the last section.

OTHER PIANO LEARNING SYSTEMS

Thanks to the popularization of new learning phenomenon recent findings indicate that AR-assisted music education seems to be a viable approach that can alleviate the shortage of qualified music teachers. Studies on this type of education have also started to gain momentum. In 2001 many studies were focused on personal contextual awareness [14]. We were invoking Cheng that develops two working systems with the focus on exploiting the user's own body parts as triggers for the augmentation. HANDEL does not have a simplified display version for those who don't know how to read sheet music, instead it makes use of the music notation itself augmented on the keyboard. Here, a see-through head mounted display is required so that the user's hands are free to interact with the keyboard. The practice session begins at first, with the loaded music score into the program where it is handled by a custom music score language and secondly with the nod of the pianist when he's ready to practice. No notes are rendered unless the user focuses on the hands

playing piano. The score is displayed at a fixed position in order to avoid confusion from seeing keys fly all over the keyboard together with the moving hand accordingly: the notes appear in the right side of the display for the right hand and similarly for the left hand. Cheng only adopted the skin color detection method given that the only thing the user will see through the HMD are the hands and the piano. Additionally, a FFT phase correlation analysis is utilized to decide whether the student's head is turned to the left or right. Another Android app, called AR Musical [15], can keep learners highly engaged in the learning activity and provoke their motivation and interest. It was implemented for the evaluation of the effectiveness of teaching musical notes to children. At the same year, Hackl proposed HoloKeys application for studying piano which is able to superimpose the keys that are to be played in an AR environment [16]. The app uses MIDI files in order to process melodies and interpret the keys together with their duration. Important is the fact that this system was implemented with the help of a fiducial marker for tracking accurately the piano keys. From a menu, the user can choose a specific song he wants to play together with its speed, a playback mode in which the user can jump to a particular position within the piece and a calibration mode where the user can adjust the position of the marker on the piano so that the optimal augmentation is displayed. Additionally, the special feature of this system is represented by the fact that musical pieces can be displayed in two different ways (the note becomes highlighted in the moment it needs to be played) and Beatmania way (the following notes can be seen in advance). In 2019, Zeng designed the FunPianoAR system [17], remarkable for its concept of paired play in which one player plays the right-hand melody while the other player focuses on the left-hand notes. Zeng also chose to use the fiducial marker tracking instead of the markerless recognition given that all keys on the piano look the same and it's difficult to differentiate them. Furthermore, the use of HMD is required again as the application runs on AR smart glasses which the user is wearing while sitting in front of the piano so that notes augmentations appear on the keyboard. The FunPianoAR system displays the keys in a unique way: a colored cube is superimposed over the note that should be played at the respective moment and a contoured cube will appear on the notes that follow to be pressed. Additionally, the application superimposes arrows to show the direction the user should look if the note that should be played is not in his sight based on the four fiducial markers on the keyboard. The few AR applications presented above focus only on superimposing instructions upon a keyboard so that the user could press the right notes in an effortless way. In 2016, an AR system which calculates a score depending on the pianist's performance in order to provide feedback was presented [18]. It is demonstrated the importance of receiving feedback, when you study to play piano. A distinct particularity for this system is represented by the way the feedback is transmitted, namely, through a rendered 3D virtual character. Moreover, the required

hardware consists of either a computer keyboard or an electronic MIDI keyboard together with a pair of Google Cardboard and a smartphone. In addition to this, the player can add MIDI files he wants to practice to the application. In the same way as the other systems, the notes will be interpreted and displayed on the screen above the keyboard so the user will know what keys to press. They will be represented by rectangles of a certain dimension, serving as the duration of a note. The evaluation is performed comparing the MIDI files initially imported with the input of the pianist during the performance. Therefore, a score is provided which is sent further to the feedback module.

Indeed, the audio and visual feedback provided by the app through AR technology can stimulate multiple learning channels simultaneously and help to develop young learners' visual [19], auditory [20], and motor skills [21]. Starting from the applications already described, our application continues to offer new ways to teach people how to play piano using AR and without needing to know or learn the music notation. There are several app features, which will be presented in the section below.

THE PROTOTYPE

Physical Setup

Firstly, the main component that is needed is a physical piano which is used for the actual playing. Above the piano keyboard, the fiducial marker is placed and is used for initial tracking. A piano keyboard with 88 keys is recommended (see Figure 1), although the application would work on any size of the keyboard.

Figure 1. The setup with the fiducial marker.

The next required component is represented by a smartphone



with Android and a Google Cardboard headset. The VR headset will take as input the video captured by the mobile camera and the augmentations will be projected into reality in front of the user. The application will run on the smartphone and will be responsible for tracking the image, displaying the augmentations, playing the sounds from the melodies and interacting with the user through an interface. The VR headset should be one with a remote such that when the user has the headset on, he could play the song by

pressing the button from the remote and not from the screen of the device.

User Interface

A simple interface was implemented in order to allow the user to interact with the system easily. The GUI includes two components, the menu and the interface which is displayed during the play game. The menu component (Figure 2) is the starting interface when the user enters the application. The interaction will be performed by pressing the buttons from the mobile screen right before the user places his device inside the VR headset. In the main menu there are multiple actions he can perform.



Moreover, the player can choose a song he wants to play from his device, which will be represented by a MIDI file. A file browser will be displayed in this scope, from which he can navigate through his folders and files in the android system. Secondly, there is a Settings button from which the player can set the tempo (the speed) of the song. If the tempo is not set, the original one will be played. This functionality will help the user to firstly listen to the original song and then he can slow down the tempo as much as he wants, so that he can press all the notes correctly, at the right timing. Lastly, there is the **exit** button which will exit the application and the play button which will bring the user into the play mode interface, where he can study the melody he chose. In the play mode interface, the user can see the title of the song he chose in the up-left corner and some action buttons in the bottom line. Afterwards, the user should detect the marker on the piano and a piano model will be rendered. The Detected button should be pressed afterwards so that the 3D model doesn't depend on the fiducial marker anymore. After the detection, the user can calibrate the model to match the real piano and play the song. A more detailed explanation about the calibration and augmentation module will be described in the implementation chapter. The last button is the Back button which can be used to return to the main menu where the user can change the tempo of the song or choose another song to learn. This way he can fluctuate in the tempos and pieces he exercises.

Calibration Module

The calibration module consists of more phases. The first one is represented by tracking the image target set in the Vuforia SDK. The application recognizes the fiducial marker and with Vuforia's help and knows the place where to render the 3D object that is mapped to the 2D image. After this detection is complete, the user uses his fingers to scale the 3D model and move it so it is calibrated with the real piano. This is performed with the help of the free Lean Touch package from the Unity Asset Store. When talking about Vuforia SDK, in Unity Editor there are only two objects that we have worked with and those are the AR Camera and the Image Target. The AR Camera is responsible for modifying the input from the mobile camera such that 3D augmented objects are displayed in 3D space. On the other hand, the Image Target is the 2D marker that must be specified so that Vuforia knows how to render the 3D model in space relative to the image. Therefore, the image target will become the parent of the displayed object, so wherever the image target is detected the object will be rendered (Figure 3).

Figure 3. The 2D image with the 3D rendered object.



In this application context, this kind of tracking is not the desired behavior because the user should be able to watch the piano keyboard and his fingers while playing and not only the marker. Therefore, once the object is tracked initially, the tracking stops and at the same time, the rendered object is placed with the same transform component as a child to an empty object that has the same position and rotation as the image target. This way we can keep the rendered object in the same position even if the marker is not in the user's visual



field anymore. Moreover, if the object's position was disrupted, the user has the option to start the tracking again for a better display, pressing the Detect button again. This time this method will be called and the rendered object will be set as a child to the Image Target again. Once the detection is done, the user will have an object rendered in space, which is independent of the fiducial marker. This object is represented by a 3D piano model (see Figure 4) which consists of 84 keys separated in 7 octaves each with 12 keys.

Figure 4. 3D Piano Model.

The piano model is a free downloaded model under an *.obj* extension and imported in Unity. The model was modified so that the unnecessary parts of the piano were removed and only the keyboard itself remained. The 3D object can be modified by the user pinching the piano to scale it, to rotate it on the X axis or to drag it across the space. Even though

the model has 84 keys, the calibration will work on any size of keyboard because the user will calibrate the model with the centered key dimension and not with the whole keyboard size.

Augmentation Module

In our opinion, this is one of the main modules of this application as it represents the visualization of the notes from a loaded song. Once the piano model is detected and calibrated, the user can press the Play Song button, which will start the visualization. Also, this action will hide the piano model used for setups so that the user can see the real keyboard on which he will press certain keys. This visualization contains 2 components: the keys and the place they are rendered on.

(1) Background plane: Firstly, a semi-transparent plane is created, representing the surface on which the notes will be displayed (Figure 5). This canvas will have the same width as the calibrated 3D piano model and a default height which will always remain the same: the height of exactly 3 notes, each note with a duration of exactly a beat. The created plane will be set as a child to the piano model which is not visible anymore, so that it can inherit properties such as position, scale and rotation. However, the rotation should not be changed as we want the canvas to remain parallel with the keyboard so that the notes can be displayed nicely, so rotation was stopped on the Z axis relative to its parent.



Figure 5. The plane the notes are rendered on.

The plane is divided into more sections horizontally and vertically. Each vertical division represents the beginning of a new octave, on each C note. Therefore, the canvas is divided for a piano of 7 octaves and if it's used on a smaller piano the grids will scale correctly after the right calibration. On the other hand, a horizontal division represents the size of a note that has the duration exactly of one beat. The lines are drawn programmatically using the LineRenderer Component and customizing each line as desired, at its correct position with its correct orientation. The whole plane is made of 3 smaller planes that are united, and on each small plane 7 lines are drawn. Each line is rendered by manipulating the vertex vector from the MeshFilter Component of a plane.

(2) Virtual piano: Once the calibration on the piano model is done, a virtual piano instance is created. This virtual piano class follows the Singleton Design Pattern so that there is only one object instantiated at runtime. The class has some attributes such as a list of all the keys, the number of black keys and the number of white keys. The list of all keys is needed so that we know how big the keyboard is, in case we want to use a smaller piano model for our 3D object. The virtual keyboard is created by iterating through all piano model's children, each being represented by a specific key. In our virtual piano, the color of each key should also be retained for faster future searches. This can be obtained by looking at each key's name such that a white key will always be formed from 2 characters (for example A7), one representing the note and the other, and the octave the note belongs to. On the other hand, a black key will always have 3 characters (for example C4#), the first 2 characters having the same signification as the white key and the last symbol being specific to the black keys only.

(3) Rendering the keys: The keys are generated after parsing the midi successfully so that they will be set as a Nested List called Notes in the Singleton class Song. More details about how these notes are read and stored in memory are found in the MIDI subchapter. The first step that needs to be taken when the keys are rendered is to hide the 3D piano model that was used for calibration earlier, so that it won't represent discomfort to the user who wants to see the real piano. Once the model is hidden, we instantiate a prefab for every note we read. Unity's Prefab is used to help us reuse the GameObject created for the white and for the black keys (see Figure 6).

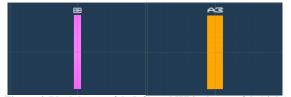


Figure 6. Black key prefab (left) and White key prefab (right).

The 2 kinds of keys are rendered visually different so that the user can recognize what kind of notes are approaching the keyboard. Each prefab consists of 2 components: a plane with a specific color and scale for Z and X axes, and a text representing the name of the note. The color pink was chosen for the black key prefab (Figure 6 left) and the yellow color was chosen for the white key prefab (Figure 6 right) as these are distinctive colors when rendered over a piano. Moreover, the Z scale is originally set so that it represents the dimension of a note that has the duration exactly of one beat and the white key prefab is slightly thicker because it has a X scale larger so that it corresponds to the proportion of the keys on the real piano. On a real keyboard, the white keys have a width of 2.3 times larger than the black keys, so in this application the white keys have the X scale of 2.3 while the black keys have the X scale of 1. Furthermore, the first time the keys are rendered, they will be displayed just above the background plane. That's the starting point on the Z axes for positioning the notes. When a note is rendered, its scale is modified relative to the duration of a note so if the note lasts 2 beats its Z scale will be multiplied with 2. This will help the user to realize how long he should press the respective

key. Additionally, the notes have an absolute time which shows the exact time in the song at which the note is played. Therefore, the next thing that is done when a key is rendered is to change its Z position relative to its absolute time. This way all keys will be rendered in the right position right from the start. Lastly, to create a key that respects our notation, we change the text from each instantiated prefab with the note's name so that the user can get used to the association between name and their position on the keyboard (see Figure 7).

Figure 7. Rendered notes approaching the keyboard.

After the notes are rendered at their precise position each with its own duration, they should start moving towards the keyboard with a set tempo. Because the tempo will be the



same during the whole melody, all notes should move with the same speed, regardless of the duration of each duration. In order to achieve effect, all notes that are rendered specific to a melody, are going to be set as children to a parent object called Notes. The Notes object is also the child of the Plane game object which contains the planes for rendering. This way, all rendered keys will be orientated and rotated automatically after their plane parent. On the other hand, the role of the Notes game object is to move all keys at once, after the Play Song Button is pressed. The strategy we applied for moving the notes with respect to a tempo is by creating and calling coroutines. A coroutine is like a function that has the ability to pause execution and return control to Unity but then to continue where it left off on the following frame. The return type of a coroutine is always an IEnumerator and in order to specify in the method where and for how long we should wait, a yield return statement must be included. Summarily, coroutines were used for structure and controlling the flow of the movement. Because we know that the plane represents exactly 3 beats and we know the tempo, we can compute the number of seconds that should pass from the upper part of the plane till the bottom part. Now we can build the movement for an iteration that lasts for the number of seconds that we've just computed. Moreover, we can estimate how many iterations a song will last as we know how long the song is. This way we can move the game object Notes, maintaining the tempo as it will always go through the same distance but between different start and end points. The movement itself is created with the Lerp method, which receives two vectors, the initial and final one and a value used to interpolate between them, and returns a new vector to which linear interpolation was applied. The interpolation value is given by the ratio between the time that has passed

since the beginning of the iteration and the total number of seconds that must muss in that iteration. The final vector is computed by unity with the following formula: $a + (b - a) \times t$, where *a* represents the starting vector, while *b* is the final one.

Another important feature in rendering the keys after the movement segment is what happens the moment when they reach the real keyboard. As we know where the bottom of the plane is, we can check every frame if a certain note has passed that point, that's the moment when the key should be played. Firstly, we will take the first child of the Notes game object and this will be checked when it reaches the end position in the plane. When that happens, the group keys will have its parents changed to the plane directly so the actual Notes object's first child will become the next group keys we have to check. A child of the Notes game object is represented by an empty object with more notes that should be played at the same time. Once a group of keys reaches the end, they should slowly disappear into the keyboard instead of going further over it. In order to make that happen, scaling was applied to each key from a group at once. Coroutines were used in this scenario as well, because it allows us to run in parallel the movement of the remaining notes together with the scaling of the actual notes that should be played. In order to maintain consistency, the notes should scale with the same speed that they were moving until now, so that the user knows how long to keep the keys pressed. We can proceed as we did before and compute the number of seconds a note should last, knowing the tempo of the melody and the duration of the note. Therefore, we can again use the Lerp function, this time for scaling, which is also represented by vectors. The major problem in this context is that the scale will proceed relative to its pivot which is positioned in the center of the note. On that account, we should move the pivot programmatically at the bottom of the figure so that the scale will look like a note going into a real key from the piano. Regarding this, as we can't actually move the pivot, we went through all vertices from the plane's mesh component and moved them instead, half the size up. This way, the pivot remained at its original position but this time this position represents the end of the note. This intermediate step is taken in the moment when all notes are first spawned, before the movement starts. The note end scale is represented by its original scale with the Z axes set on 0. When a note finishes its scale it means that the note should not be played anymore and it is automatically destroyed.

MIDI and Audio

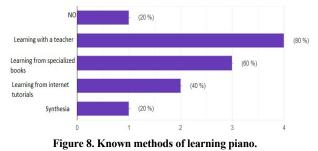
Once the player chooses a melody, the MIDI parser comes into action, which parses the respective file so that we can visually represent all the notes. The second component, the Audio Manager, is used the moment when a note reaches the end of the plane, so that it will make a sound from that specific key.

(1) Midi Parser: In order to parse the MIDI files, we used the NAudio library for C# and Unity. There are two types of MIDI files that we should be aware of. The first one is when the piano song is represented in the file with notes for both the right hand and the left hand, while the second one represents all the notes from the song for a single hand, even though the melody itself is for both hands. In this regard, we have created a Song class with a List called Notes which will contain more lists, each one representing a group of notes that begin at the same absolute time. This way we can memorize any kind of MIDI representation, because it is based on the absolute time of a specific note, which represents the time in the song that the note is played. Another feature we parse from the file is the details about each note such as the note's name, its duration and velocity, representing the volume to each note it's played. The note's name required a little bit more attention because the library returned each note an octave higher than it was. This means we needed to modify each note's octave manually, so eventually each note's name. The black note is differentiated from the white notes by termination such as "b" or "#", and some black keys can have two names: one for each symbol. For example, the key *Db* can also be called C# or D# can be *Eb*, because of these differences we tried to normalize the naming to only one version so that it will be the same in the whole application. We went for the standardized version we learnt such that for the examples above we would choose the C# and the Eb.

(2) Audio Manager: For creating the audio manager we made use of the Audio library in Unity. An empty game object was created to which we added manually and programmatically more Audio Sources Components. An audio source component contains an audio clip which holds an *mp3* sound. In the first place, the main 12 sounds from the 4th octave of an 88-keyboard piano, were added manually: the sounds from C4, C#4 to B4, Bb4. Afterwards, and all the other Audio Source Components were added automatically in the Awake method which is called before the Start method. We created the same sounds for all 7 octaves changing only the pitch of the main sound. Notes are separated by the factor 21/12 or 1.059463, so we should compute the pitch for the remaining notes and starting at any note the frequency to other notes may be calculated from its frequency by: Freq = note \times 2 N/12, where N is the number of notes away from the starting note. N may be positive, negative or zero.

EVALUATION AND INTERPRETATION

We collected opinions about the *Teacherless Piano AR* from end-users in order to understand the impact of our application together with improvements that can be done in the future. Usability tests were done in-person for both, females and males, within different age ranges (between 18 and 50 years) with different piano background and AR experience. In the first place, 3 out of 5 people do not have experience with playing piano, meanwhile one of them learned a bit before and the other one has an intermediate level. On the other hand, when speaking about the AR experience, we have users with all kinds of experience level (between 1-less experience and 5-most experience), from not knowing about this concept to really working with it on personal projects. The conducted usability tests consisted of an **introduction** about the application's scope, a **short demo** in which we showed the main features followed by an **attempt to use it** the same way by the user and a **post-test questionnaire**. The users needed a smartphone on which we installed the app through an APK file. They were asked if they knew any other methods of learning piano, or if they were familiar with AR method beforehand (Fig. 8).



Clearly, the most known method was learning piano with the help of a professor, followed by individually learning from specialized books. Two users thought of the possibility of learning piano from internet tutorials, only one knew about the Synthesia game, and one didn't think of any method. Also, this experiment showed that our method was a completely new one for all. Afterwards, a demo was presented with the main features. Obviously, for the setup to be complete the user needed a real piano on which to test the application and a VR set to place their mobile and experience the AR part. Even though we explained and helped them adjust the VR set. It was a complicated phase because most of them have not used one before (80%). Firstly, the menu was introduced, where a user can select a song from the device and a settings part where the user can change the tempo of that piece. For this part, we copied some MIDI files which represent some songs, into the user's device beforehand so that he has something to choose from. Then, to choose the correct tempo, the piece should be played once or twice for testing. For the sake of testing fast, we gave them the ranges of tempo for each melody, the tempo in which the song is slow and the normal tempo so that the user could choose the right speed. After the menu was presented, the users had to answer some questions about its usefulness and how intuitive it is (between 1-less intuitive and 5-most intuitive), as well as some suggestions about how to make it better. Most users considered the menu very useful and intuitive (80%), but some of them gave the score of 4 alongside some suggestions (Table 1). All these proposals suggest the user's need to have a 3D piano rendered model more dynamic.

User suggestions	Female	Male
The application needs a built-in tutorial.	Х	

Proceedings of RoCHI 2021

Setting language preferences	Х	
An option to choose a method to retain all the songs that were once introduced in the application	Х	Х
A more sophisticated library where the user could add, remove or choose songs directly from there when they want to learn it	Х	X
An option where the user could select the number of keys his piano	Х	Х
An option to provide feedback when the user has played the note correctly or not.	x	Х

Table 10. User suggestions.

Afterwards, during the demo presentation, the main part was explained including the calibration of the 3D piano model with the real keyboard and the play section where we showed them how to follow the notes that go down onto the keyboard. In regard with the calibration section, the detection of the marker was described, then we let them do that by themself and asked about the difficulty of this technique (see Figure 9).

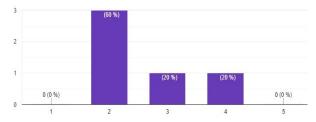


Figure 9. The difficulty of piano calibration, between 1-less difficult and 5-most difficult.

As the graphic shows, most users found it almost easy, but in the case of the people who were aged 40 years old and over, it seemed more difficult for them to control. The difficult part was represented by moving the piano 3D model to its correct position after the detection took place. Basically, each player chose a song and tried to play it fully from beginning to the end in more rounds. They were asked how well they could follow the piece. The proportion on the scale (between 1-less difficult and 5-most difficult) was: 60% moderately difficult, because the virtual reality experience would make them feel a bit dizzy and 40% difficult, because they couldn't get all the notes right. The users mentioned that it was also a bit hard to watch the hands and the notes at the same time. This problem appeared because the users didn't have experience with piano so they would always pay attention at their hands on the keyboard. Among the suggestions about the options in the middle of the game, a provided feedback from the application was the most requested so that the users could see if they played the right note or if they missed it.

This could be done by coloring the note in red if it is missed or green if it is right. Also, a score system could be implemented so the user would always try to beat his own score and get better every time. Another suggestion in the same direction was made, but this time the user mentioned that the melody shouldn't continue unless the right key was pressed. This could be represented by another module, a Training Module, where the user could control the melody and the tempo wouldn't be given in the beginning. A pause button is also requested in the middle of the song so that the player can analyze the notes that would come next, or even an option where the user could change the tempo in the middle of the game. In the end, after playing and testing the application a bit, each user was asked if a piano song could be learned using this method, and all of them answered "YES". They considered it the most useful and interesting approach. In conclusion, the application can be useful and there are more improvements that can be done to this prototype resulting from this survey. Almost all users were able in the end to keep up with the whole song, but it wasn't easy for all of them as they were inexperienced in piano. Not many users were chosen for this test because of the complex setup that has to be done in the beginning, but those who tested it, remarked its potential and were pleased with it. The suggestions they made are acceptable to be taken into account, but only at user experience level because most of them were not technical people and the concept of augmented reality was quite innovative to them. As it can be seen in the graphic below (see Figure 10), most users managed to go through the application by themself eventually, only one person said it was a bit uncomfortable, mainly because of all the steps that needed to be taken in order to start the song.

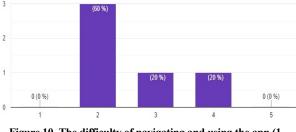


Figure 10. The difficulty of navigating and using the app (1difficult, 5-not difficult).

CONCLUSIONS

In order to reach the goal to teach people to play piano using AR, the keys that should be played are augmented so that the user knows the note that must be pressed. First, the main component that is needed is a physical piano which is used for the actual playing. Above the piano keyboard, the fiducial marker is placed and is used for initial tracking with Vuforia. The next required component is represented by a smartphone with Android and a Google Cardboard headset. The VR headset will take as input the video captured by the mobile camera and the augmentations will be projected into reality in front of the user. The application will run on the

smartphone and will be responsible for tracking the image, displaying the augmentations, playing the sounds from the melodies and interacting with the user through an interface. The VR headset should be one with a remote such that when the user has the headset on, he could play the song by pressing the button from the remote and note from the screen. Moreover, the GUI includes two components, the menu and the interface which is displayed during the play game.

Future work includes on the one hand solving a part of the improvement suggestions recommended by those who tested the application and on the other hand testing with other AR/VR devices, in order to find an optimal configuration for using the application.

ACKNOWLEDGMENTS

This work was supported by project REVERT (taRgeted thErapy for adVanced colorEctal canceR paTients), Grant Agreement number: 848098, H2020-SC1-BHC-2018-2020/ H2020-SC1-2019-Two-Stage-RTD.

REFERENCES

- 1. Higgins, L. Community music: In theory and in practice. New York: *Oxford University Press*. (2012).
- 2. Macdonald, R.A.R. Music, Health, and Well-being: A Review. *International journal of qualitative studies on health and well-being*, 8 (2013).
- Weing, M., Röhlig, A., Rogers, K., Gugenheimer, J., Schaub, F., Könings, B., Rukzio, E., and Weber, M. P.I.A.N.O.: Enhancing Instrument Learning Via Interactive Projected Augmentation. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication* (UbiComp '13 Adjunct). ACM, New York, NY, USA, (2013), 75-78.
- 4. Chow, J., Feng, H., Amor, R., Wünsche, B. Music education using augmented reality with a head mounted display, (2013), 73-79.
- Das, S., Glickman, S., Hsiao, F.Y., Lee, B. Music Everywhere – Augmented Reality Piano Improvisation Learning System, (2017).
- Piano Wizard Academy, Piano Wizard, Software, US, (2005).
- 7. Yousician. Yousician. Web & Mobile. Finland, (2014).
- 8. The ONE Smart Piano, The One Light, Hardware, China, (2016).
- 9. Azuma, R.T. A survey of augmented reality. Presence: Teleoper. *Virtual Environ.* 6, 4 (1997), 355-385.
- Chițaniuc, M. and Iftene, A. GeoAR An Augmented Reality Application to Learn Geography. In *Romanian*

Journal of Human-Computer Interaction, 11, 2 (2018), 93-108.

- 11. Macariu, C., Iftene, A. and Gîfu, D. Learn Chemistry with Augmented Reality. In 24rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, 16-18 September, Procedia Computer Science, 176 (2020), 2133-2142.
- 12. Păduraru, B. M. and Iftene, A. Tower Defense with Augmented Reality. In *Proceedings of the 14th Conference on Human Computer Interaction - RoCHI* 2017, Craiova, Romania, (2017), 113-118.
- Pinzariu, M.N. and Iftene, A. Sphero Multiplayer Augmented Game (SMAUG). In *International Conference on Human-Computer Interaction*, Iasi, Romania, (2016), 46-49.
- 14. Cheng, L.T. and Robinson, J. Personal contextual awareness through visual focus. In *Intelligent Systems, IEEE*, 16 (2001), 16-20.
- 15. Lemos, B. H. V., Corrêa, A. G. D., Nascimento, M. D., and Lopes, R. D. D. Augmented reality musical app to support children's musical education. In *Comput. Sci. Inf. Technol.* 5 (2017), 121-127.
- Hackl, D. and Anthes, C. HoloKeys An Augmented Reality Application for Learning the Piano. In *Forum Media Technology*, (2017).
- 17. Zeng, H., He, X. and Pan, H. FunPianoAR: A Novel AR Application for Piano Learning Considering Paired Play Based on Multi-Marker Tracking. In *Journal of Physics: Conference Series*, 1229, 012072 (2019).
- Fernandez, C., Paliyawan, P., Chu, C. and Thawonmas, R. Piano Learning Application with Feedback Provided by an AR Virtual Character, (2016).
- Di Serio, Á., Ibáñez, M. B., and Kloos, C. D. Impact of an augmented reality system on students' motivation for a visual art course. In *Comput. Educucation* 68 (2013), 586-596.
- 20. Bauer, V., Nagele, A., Baume, C., Cowlishaw, T., Cooke, H., Pike, C., et al. Designing an interactive and collaborative experience in audio augmented reality. In *Virtual Reality and Augmented Reality Lecture Notes in Computer Science*, Cham: Springer International Publishing (2019), 305-311.
- 21. Chang, K.E., Zhang, J., Huang, Y.S., Liu, T.C. and Sung, Y.T. Applying augmented reality in physical education on motor skills learning. In *Interact. Learn. Environment* 28 (2020), 685-697.