InvisibleSound: An App Enabling Blind People to Compose Music

Maximilian Stolze, Helmut Hlavacs

InvisibleSound: An App Enabling Blind People to Compose Music

Maximilian Stolze and Helmut Hlavacs*

University of Vienna, Research Group Entertainment Computing
Vienna, Austria,
maxstolze@yahoo.de, helmut.hlavacs@univie.ac.at

Abstract. InvisibleSound is a mobile application software developed for blind people to compose music. The app's handling is adapted to the needs and requirements of blind people. As a control mechanism to navigate through the app's functions we tested two modes: a common swipe control and an interactive motion control by moving the phone. Also publishing and sharing the composed songs on a shared server is possible. The app was created with Unity and is available for Android and iOS.

Keywords: Blind · Compose music · Creativity tool

1 Introduction

Nowadays there are hundreds of music apps and hundreds more that support to create, edit and publish own songs. All of these apps have one thing in common: their complexity. They mostly provide a huge amount of functions, controlled and used by numerous buttons and sliders. This is especially true for many popular music apps, which often provide interfaces not fit to be used for people with bad eye sight or even being blind. The latter usually depend on screen readers, which in the presence of the above GUIs cannot be used by them at all. So the Question was: How could a music composing app be designed to be accessible for blind and low-vision users?

In order to answer this question we created the music composing app InvisibleSound. The app's handling is adapted to the needs and requirements of blind people. Two different control mechanism to navigate through the apps functions are provided: a common screen swipe control and an interactive motion control by moving the phone (see below). With two different control techniques, the app gives blind and low vision users the opportunity to choose their individual favourite handling for apps.

2 Related Work

As mentioned there are a lot of composing apps and software and also a few with some support for screen readers. But by now there is no composing app for blind

users in particular. Nevertheless the field of entertainment computing focused a lot on accessibility features and human computer interaction for people with disabilitiesfocusing on game design and educational games [4]. There are some guidelines for that, helping to design software for people with disabilities and give the developers some assistance [5, 7].

It is possible to categorize the different types of Impairments [9]: 1. hearing impairment, 2. motor impairment, 3. cognitive impairment and 4. visual impairment, including bad eye sight, complete blindness and color blindness. This paper we focus on the last impairment, including all three types. Also there are already some music games for blind users like Finger Dance [3] and a blind version of the game Guitar Hero, called Blind Hero [8].

In some cases it is possible to replace visual components with audio, haptic or just enhance visuals (if the user is not totally blind) [9, Table 4]. But there is still a small market for accessibility games because there are still two main arguments: "The cost of implementing accessibility features isn't worth the return" [10] and "There isn't a wide-enough audience to make accessibility development worthwhile" [10]. But these arguments are outdated, nowadays it is worth to invest in accessibility features and also the costs decreased over the last years [6, 10].

There are several ways to make such games. One way is to create new platforms like the "Blindstation"[2] that was realized by the "TiM Project" (tacticle Interactive Multimedia)[1]. This project was formed by the European commission and tries to make games accessibly for children between 3 and 10 years old with vision rates "less than 0.05" [1].

3 Handling and Control

The app provides two selectable types of handling: a screen swipe handling and a motion handling. The swipe control provides all options by swiping horizontal and vertical over the screen.[1] Swiping horizontal changes between the different options in every single menu. Every time the option will be announced verbally to the user. Swiping vertical offers the option to go back to the menu before. So the user does not have to do unnecessary swipes vertically, just to get the *back option*. A single click on the screen confirms the active selected option. When an option is selected also a haptic feedback in form of a short vibration is given, so the user knows that his click was successfully recognized by his device.

The motion control the user needs an image that can be downloaded from `http://homepage.univie.ac.at/a1308624/invisiblesound/index.html`, and seen in Figure 1.

[1] `https://youtu.be/1HbfuwVbFWM`

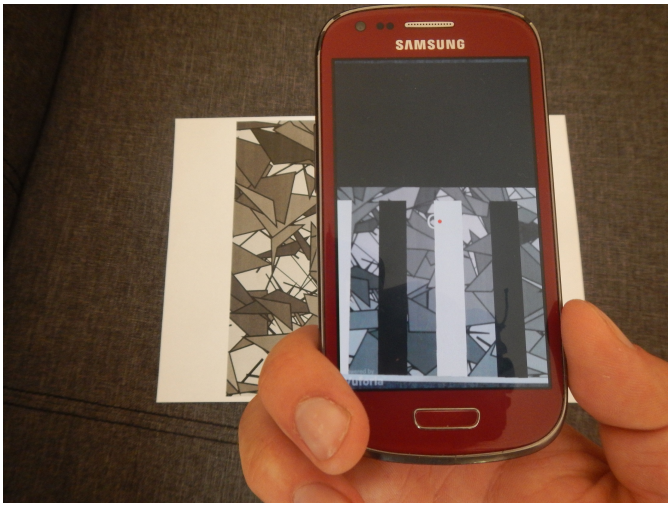

Fig. 1. Picture an Android Device running the app with the focusing on the Vuforia reference image and showing the augmented reality objects,

This target image is used by the app with Vuforia, an augmented reality SDK. Once the camera records this target image, Vuforia is able to detect it automatically, and with it, any motion of the mobile phone. Moving the mobile phone now up or down, left or right then triggers the same control actions as would be triggered by swipe moves, as described above.

InvisibleSound is grouped into three *modes*. the first is the *move mode*, the second the *swipe mode* and third is the *network mode*. Move and swipe mode have the same main functions that are required to compose and edit own songs. They just differ in their handling (*swiping* and *moving the phone*), as described in Section 3.

The network mode is responsible for all online components of the app, for example publishing a song on the server. The users can choose between these three modes in the first menu directly after starting the app. It is always possible to change the mode by just going back to the *first start menu*.

3.1 The Composing Process

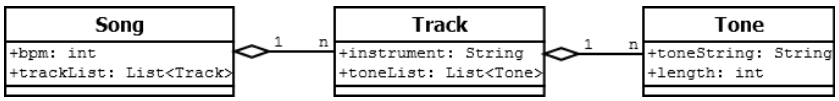

Fig. 2. Class Diagram: Song

A song (Figure 2) has a playback speed, called *beats per minute (bpm)*. The bpm declares the delay time of a note. For example in a song with 60 bpm the delay time of a quarter note is exactly one second. This can be resolved with the following formula:

$$\boxed{\begin{aligned} time &= (60/bpm) \times 4s \\ noteTime &= (time/toneLength) \end{aligned}}$$

Here, *time* denotes the delay time of a full note with the current bpm in seconds, *noteTime* is the actual time of the tone in seconds, *toneLength* can have the values of 1 (full note), 2 (half note), 4 (quarter note) and, 8 (eighth note)
Example 1: Song with 60 bpm and a quarter note ($toneLength = 4$):

$$\boxed{\begin{aligned} time &= (60/60) \times 4\,s = 4\,s \\ noteTime &= (4\,s/4) = 1\,s \end{aligned}}$$

Example 2: Song with 120 bpm and a quarter note ($toneLength = 4$):

$$\boxed{\begin{aligned} time &= (60/120) \times 4\,s = 2\,s \\ noteTime &= (2\,s/4) = 0.5\,s \end{aligned}}$$

A song has also a list with objects of the class Track. A track object needs two attributes: an instrument and a list with all tones this instrument plays in the song. So each track represents an instrument in the song. A tone object then represents a note, played by the instrument. The *toneString* is the sound itself saved as a string. The *length* is the type of the note (whole note, half note, etc.).

Therefore, a song has its bpm, its list of tracks, each track representing exactly one instrument and having a list of tones, with relative duration, depending on the bpm of the song. Songs are saved as text-files in the *application persistent data path*.

After selecting the *move-* ore *swipe mode* and with this the kind of handling the user wants to create a song, the next menu is the *start menu* of the mode itself. Here the user has three options: Play the song, edit the song, a create a new song.

Selecting **create song** brings the user to the first step in the composing process:

Song menu: In the *song menu* all options to edit the specifications for the whole song are provided: Change bpm, play the song, add a new track, edit a track, save the song, and go back.

Instrument menu: In the composing process the next step is to add a track to the new song. When picking this option the user comes to the *pick instrument menu*. Here the instrument for the track needs to be selected. The next menu is the *Instrument menu*. Here the user has all functions and options to create and

edit a single track. A track needs tones and so the next option in the process is to add tones. This happens in the *add tone menu*.

The user can go through all available tones and sounds of the instrument and can add them to the track. After every tone selection the user needs to declare the type of the note (full, half, quarter, eighth). As mentioned is there a *back* option in every menu, so the user just needs to go back to the upper menu. To save the created song, the user picks the *save song* option and comes to the *select- memory slot menu*. Is the selected memory slot already in use the user is asked to overwrite. If not, he can pick another memory slot. The last version of the app provides ten memory slots to save songs locally on the device. After saving the song, the user is back in the *start menu*.

Network Menu The *network menu* will be entered from the first *start menu* when selection the *network mode*. This menu has all functions that are necessary to upload own songs, listen to songs of other users and also to own uploaded songs. To enter the *network mode* and the *network menu* an Internet connection is required. This mode is controlled with the swipe handling. When a user enters the mode the first time the app creates a connection to an online database and the user gets a unique ID. With this ID the user can be identified in the network. When this process is done the user has the following five options: announce an ID, upload a song, play newest song, search for a song id, or play own online songs. Now the most important design choices have been explained and also this menu layout was tested successfully at the second user test (see section 4).

4 Evaluation

By now the app was tested two times by blind children at the Bundes-Blinden-erziehungsinstitut in Vienna, first together with one blind expert, in a second test with 10 blind children in a workshop setting.

For blind people it is easier and more in common to manage a space directly in front of them, for example the half square meter in front of them on a table. The image target should now be placed on a table in front of the user. So since the first evaluation the objects needed to be replaced to fit to the moves from left to right instead of up and down. The second big innovation that resulted form this first test was the swipe control. We learned that blind users are very in common with swipe moves and switching between options with this kind of handling.

For the second test all options were provided with both control mechanism and the network functions. Here we could see how some blind people use their phone in general: holding the device with the speakers on their ear with very quiet sound setting and swiping through the options. For this they hold the device in in one hand and upright. So we need to block the apps auto rotation, otherwise the app will rotate and the swipe direction would change.

All in all the students were very excited and agreed to test the app again when the bugs are fixed.

5 Conclusion

The app is something new on the market. Till now there is no composing app only for blind people. So there was no specific app to get some ideas for InvisibleSound. Only the main structure of creating a song is similar to common composing apps: A song has tracks, representing instruments and a track has tones. Also the first prototype had only the motion control and after a first user test the idea and the motivation for an alternative swipe control was developed.

References

1. Archambault, D.: The tim project: Overview of results. In: International Conference on Computers for Handicapped Persons. pp. 248–256. Springer Berlin Heidelberg (2004)
2. Archambault, D., Olivier, D.: How to make games for visually impaired children. In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology. pp. 450–453. ACE '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1178477.1178578
3. Daniel Miller, Aaron Parecki, S.A.D.: Finger dance: a sound game for blind people. Assets '07: Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility pp. 253–254 (2007)
4. Hughes, K.: Adapting audio/video games for handicapped learners: Part 1. Teaching Exceptional Children pp. 80–83 (1981)
5. Siebra, C., Gouveia, T., Macedo, J., Correia, W., Penha, M., Silva, F., Santos, A., Anjos, M., Florentin, F.: Usability requirements for mobile accessibility: A study on the vision impairment. In: Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia. pp. 384–389. MUM '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2836041.2841213
6. Torrente, J., del Blanco, Á., Serrano-Laguna, Á., Vallejo-Pinto, J.Á., Moreno-Ger, P., Fernández-Manjón, B.: Towards a low cost adaptation of educational games for people with disabilities. Computer Science and Information Systems 11(1), 369–391 (2014)
7. W3C: Web content accessibility guidelines 2. https://www.w3.org/TR/WCAG20, (09.07.2016)
8. Yuan, B., Folmer, E.: Blind hero: enabling guitar hero for the visually impaired. Assets '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility pp. 169–176 (2008)
9. Yuan, B., Folmer, E., Harris, F.C.: Game accessibility: a survey. Universal Access in the Information Society 10(1), 81–100 (2011), http://dx.doi.org/10.1007/s10209-010-0189-5
10. Zahand, B.: Making video games accessible: Business justifications and design considerations. https://msdn.microsoft.com/en-us/library/ee415219.aspx, (09.07.2016)