DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-300

Musical Instruments Simulation on Mobile Platform

***Xunyu Pan*[1], *Jacob Wilson*[1], *Megan Balukoff*[1], *Anyi Liu*[2], *and Wenjuan Xu*[1]**
[1] ***Department of Computer Science and Information Technologies, Frostburg State University, Frostburg, Maryland, USA***
[2] ***Department of Computer Science, Indiana University-Purdue University Fort Wayne, Fort Wayne, Indiana, USA***

Abstract

Mobile devices are increasingly used for entertainment applications due to their ability to transcend economic and cultural barriers. One promising trend in developing mobile musical application is to simulate mobile devices as various musical instruments, rendering them suitable for real-time music playing and editing. To address some existing challenges, we introduce a novel mobile musical system aiming to simulate the functionality of musical instruments and visualize the interaction between simulating devices and mobile users. In practice, we create a friendly user interface on the touch screen of mobile devices to simulate a string or keyboard instrument. The action of finger touch is also visualized on the touch screen in a graphic format. The fusion of music playing and visual responding provides users tactile feedback and better playing experiences. We also optimize the musical algorithm to reduce the audio generation latency by minimizing system computation. The ultimate goal of the mobile musical system is to demonstrate the feasibility and the potential for converting mobile devices into a particular musical instrument. The described system plays an important role in not only serving as a powerful tool for personal entertainment, but also assisting individuals interested in learning and editing electronic music.

Introduction

The rapid growth of mobile technologies has seen the number of mobile devices (e.g. mobile phones, tablets, and netbooks) surpass that of many traditional computing devices. Meanwhile, the technology development for creation [1], transmission [2], and verification [3, 4, 5, 6] of various types of multimedia data (e.g. audio, image, and video) makes them the extremely popular applications on Internet. More recently, mobile devices are increasingly used to support numerous of entertainment applications due to their ability to transcend economic and cultural barriers. Comparable to those of their desktop counterparts, the ever-growing computation and storage capabilities of mobile devices makes it possible for many musical applications to run on different mobile platforms. The ubiquity and portability of mobile devices also compensate for their processing limitation on acoustic bandwidth and quality, making it feasible to play and learn music almost anytime and anywhere.

Currently, one promising trend in developing mobile musical application is to simulate mobile devices as various musical instruments, rendering them suitable for real-time music playing and editing. While some important progresses have recently been made on the musical creation and analysis, the development of reliable real-time musical instruments on mobile platforms is still faced with challenges from different perspectives. Although popularly deployed and used for application development on mobile devices, Android mobile operating system is not designed for supporting real-time audio applications and hence the audio generation latency affects the system performance shortly after a user starts the music playing. Moreover, due to the lack of the physical keys on mobile devices, no tactile feedback can be sent to users, making the playing experiences less desirable for users. In addition, because many existing computer music applications are proprietary, and because many of their implementation details are encrypted, it is hard to precisely determine how they operate. Consequently, the analysis and extension of such systems has become extremely difficult or even impossible.

To address these issues, we introduce a novel musical system implemented on mobile devices aiming to simulate the functionalities of musical instruments and visualize the interaction between simulating devices and mobile users. In practice, the mobile musical system is developed using the Java language on Android platform with the support of Csound [7], an audio synthesis system released under LGPL. First of all, with the touch screen supported by most mobile devices, we create a friendly user interface (UI) to simulate a string or keyboard instrument (e.g. guitar, fiddle, and piano). Second, the action of finger touch is visualized on the touch screen in a graphic format to reinforce the physical interaction between the user and the mobile device. The fusion of music playing and visual responding on the touch screen provides tactile feedback and better playing experiences for mobile users. Finally, we optimize the musical algorithm by following the API low-latency techniques and further compile the application with the assistance of ProGuard [8] tool designed specifically for Android apps. These efforts help to reduce the response time of our musical system by minimizing the computation required for sound generation. We expect that, by connecting to Internet via Wi-Fi or cellular network, the mobile musical system can serve as a useful tool for creating social communication experiences in our ever-expanding world of social networking. Two rounds of experiments are conducted to validate the reliability of the proposed musical system. In the first round, we quantitatively evaluate the performance of the musical system with several sets of music pieces to test the audio generation latency. The experimental results show that the proposed system greatly reduces the latency for audio generation when comparing to that implemented with the original Android audio stack. The musical system is then qualitatively evaluated by playing with several sets of well-known pieces of music on the latest Samsung and Google phones which are all supported by Android. In a survey conducted with the people unware of the existence of this type of musical applications, most participants provide positive feedback on both the sound quality and the interactive experiences. The described mobile instrument system plays an important role in not only serving as a powerful tool for personal entertainment, but also assisting individuals interested in learning and editing electronic music.

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-300

Related Work

The technologies of music composition on home computers have been studied by numerous of researchers for a long time since 1980s when Musical Instrument Digital Interface (MIDI) was introduced. More recently, research efforts have been focused on the exploration of computer music applications particularly designed for the real-time integration of audio and graphics [9]. Major research topics on these applications include audio visualization, game-like interface, and musical instrument on either desktop or mobile platforms.

Audio visualization is a category of computer music application who seeks direct mapping between audio signal and the corresponding graphic rendering. The real-time visual response to audio signal usually sparks people's interests in listening to music on computing devices. The *Audicle* [10] is audio programming system with deep integration between real-time visualization and the working process of the *ChucK* audio programming language [11]. An essential part of *Audicle* is the *sndtools* [12], a set of cross platform tools for simultaneously displaying related audio and visual information in real-time. *Converge* visualizer [13] is another audiovisual composition tool employed to be a blender of hundreds of images and associated daily life data collected using mobile phones from users. Another category of computer music applications involves the design of game-like interface. In *Non-Specific Gamelan Taiko Fusion* [14], high- quality samples of a specific set of drums and bells are generated by incorporating original acoustic instruments. *LUSH* [15] is a music sequencer that allows users to create music by leveraging playful visualization and organic interaction. This system offers various functionalities including flocking simulation, score file analysis, nondeterministic finite automata, and visualization of musical notes by behavior. The final category of computer music applications aims to simulate various musical instruments on mobile platforms. Early exploration of this topic involves musical creation using location information on wireless ready (e.g. WiFi or Bluetooth) [16, 17], GPS based [18, 19] or camera integrated [20] mobile devices. The concept is called "location music" [21] or "location media" [22], where the distributed location plays a conceptual role during the process of musical creation. For more advanced mobile devices, the user controllability is further enhanced by the introduction of touch screen interface. Some pioneer works employ a port of Pure Data (PD) for musical instrument simulation on Personal Digital Assistant (PDA) systems [23, 24]. *Ocarina* [25] designed for the iPhone, presents a flute-like physical interaction using microphone input, multi-touch, and accelerometers. *Magic Fiddle* [26] combines physical metaphors of a violin with the virtual elements of a game and personal music teacher on iPads.

As described above, some important progresses have been made on the development of reliable real-time musical instruments on mobile platforms. However, several critical problems still need to be addressed. As one of the most popular mobile operating systems, Android supports the development of many musical applications. However, Android based mobile applications on musical instrument simulation have not been thoroughly studied so far. This is partially due to the relatively long response time for Android audio stack which affects the system performance shortly after a user starts the music playing. Meanwhile, the tactile feedback is limited as most mobile devices do not support physical keys, which is not desirable for users. In addition,

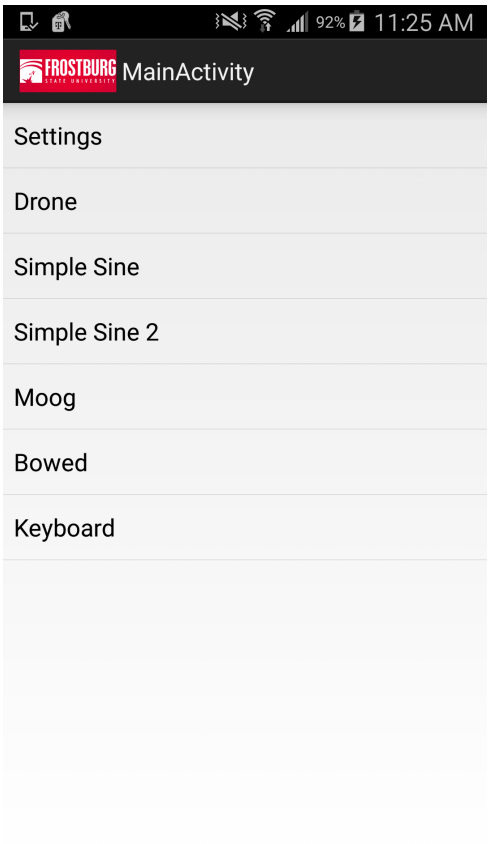

Figure 1. *The main User Interface (UI) of the proposed musical system.*

the analysis and extension of many mobile musical systems on the market are extremely difficult or even impossible largely due to the copyright issues involved with most of these commercial products.

Methods

We introduce in this section a novel real-time musical system implemented on mobile devices to simulate various musical instruments. The proposed system is specifically designed to visualize the interaction between simulating devices and mobile users. Our musical system consists of two major functional components: (a). A friendly user interface (UI) is created to simulate a string or keyboard instrument. The design is based on the touch screen technologies supported by most of the today's mobile devices. (b). The visualization of finger touch action is realized in a graphic format to enhance the physical interaction between the user and the mobile device. By the combination of music playing and virtual responding on the interface of touch screen, the tactile feedback to users provides much better playing experiences. In addition to the main functionalities, the musical algorithm of the proposed system is optimized by using the ProGuard [8] tool particularly designed for Android mobile operating system and by following the API low-latency techniques. Both of these efforts minimize the computation during the process of sound generation and hence greatly reduce the audio generation latency of the pro-

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-300

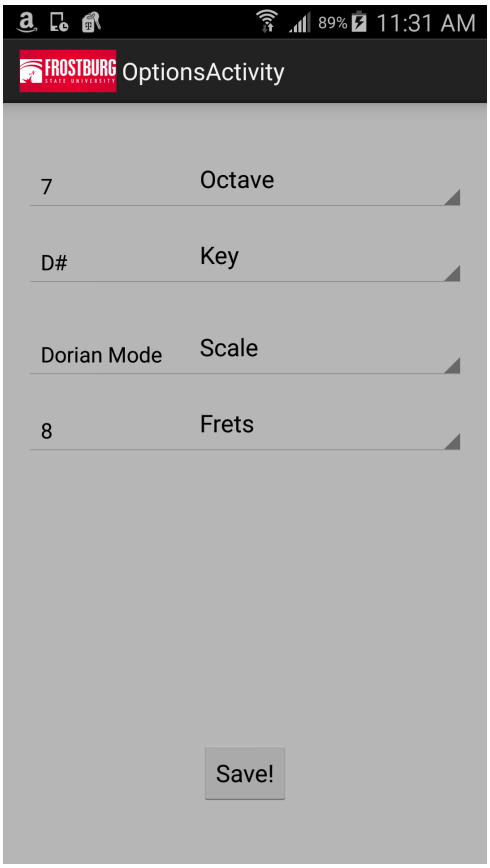

Figure 2. *The* Settings Operation *allows users to alternate various musical parameters used in the* String Mode.

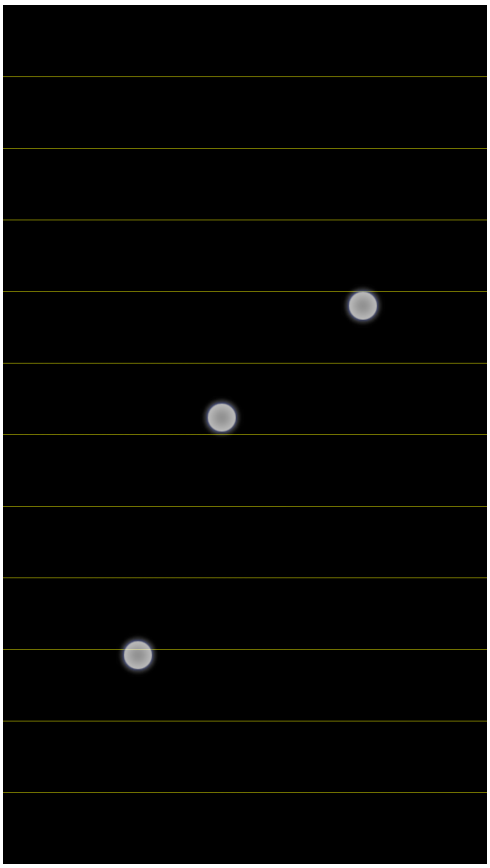

Figure 3. *The string instrument is simulated by the proposed musical system. The ripple-like effect provides important system feedback for finger touch activities when a user plays music.*

posed musical system.

User Interface

The main user interface of the our mobile musical system as shown in Figure 1 is featured with the option for users to choose from either *String Mode* (default) or *Keyboard Mode*. The *String Mode* is used to simulate the fretted musical instruments while the *Keyboard Mode* is used to simulate the keyboard musical instruments:

1. **String Mode**: In this mode, the musical system simulates a string musical instrument, such as guitar or fiddle. All frets of the instrument are displayed on the screen. The *Settings Operation* alternates four musical parameters (described below) for the string instrument. A specific sound is produced based on the combination of these four parameters.
2. **Keyboard Mode**: In this mode, the musical system simulates a keyboard musical instrument, such as piano or keytar. The keyboard layout of the instrument contains one complete octave, where the Chromatic scale notes start in C of the sixth octave.

The *Settings Operation* as shown in Figure 2 is associated with the *String Mode*. This operation allows for the alteration of four musical parameters for a string instrument: (a). *Octave*, (b). *Key*, (c). *Scale*, and (d). *Frets*. The alteration of the musical parameter *Otave* starts with 0 and ends with the 9th octave, spanning in the Chromatic scale of 108 possible musical notes. The alteration of the musical parameter *key* allows for setting the layout of all musical keys to be represented. The alteration of the musical parameter *Scale* allows for Chromatic, Major, Harmonic Minor, Natural Minor, Lydian Mode, Aeolian Mode, or Dorian Mode to be generated for a specific string instrument. The musical parameter *Frets* specifies that the number of frets displayed on the screen has the range from 5 to 18. This parameter allows for adding or cutting off frets based on their positioning. For example, a user can choose the 6th octave, key of C, Chromatic scale, and 14 frets. The canvas is hence drawn with 14 frets containing a full Chromatic scale of notes starting at C with another 2 notes in the Chromatic scale of the next octave starting again with the root note C.

When the musical system is set in the *String Mode*, the frets are generated as a set of horizontal yellow lines on the canvas at the center of the screen as shown in Figure 3. Based on the x and y coordinates of finger touch point on the 2-D canvas, the corresponding sound is generated by Csound API. The *Settings Operation* is associated with the *String Mode* to create various music playing modes. A user can create a preferred musical instrument setting by choosing a specific combination of the four musical pa-

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-300

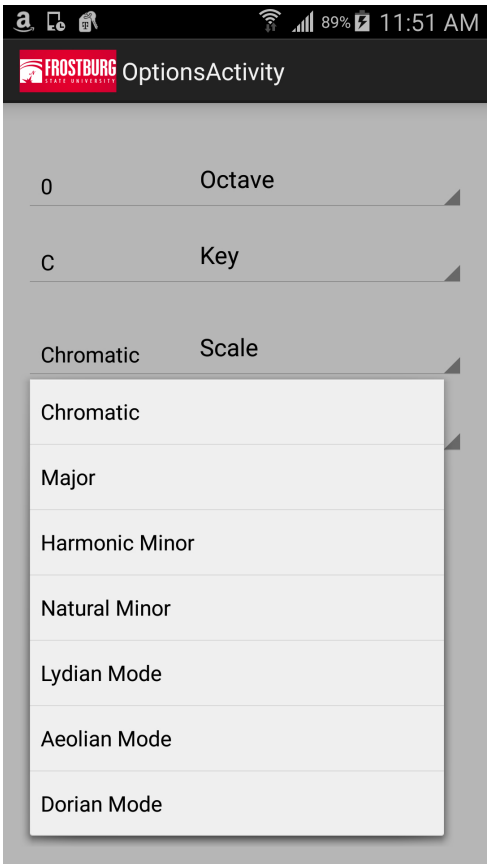

Figure 4. *The* Scale *setting in* Settings Operation *provides various* Scale *options for user selection.*

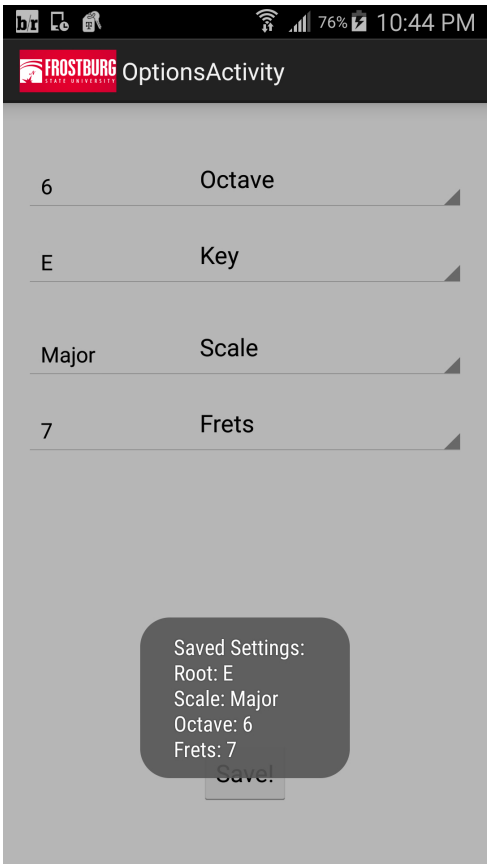

Figure 5. *The preferred musical instrument setting chosen by a user is saved for future usage.*

rameters from various mode options. For example, as shown in the Figure 4, various *Scale* options are available:

1. **Chromatic**: A 12-note scale spanning all the tones of the octave.
2. **Major**: A scale consisting of a series of whole steps except for half steps between the third and fourth and seventh and eighth degrees.
3. **Harmonic Minor**: A scale same as the natural minor scale except that the seventh degree is raised by one semitone.
4. **Natural Minor**: A scale same as relative major scale except it is built starting from the sixth note of the relative major scale
5. **Lydian Mode**: A scale starting at a root note, then comprising three whole tones, a semitone, two more whole tones, and a final semitone.
6. **Aeolian Mode**: Same as the natural minor mode.
7. **Dorian Mode**: A scale with a minor third and seventh, a major second and sixth, and a perfect fourth and fifth.

Other setting options include the *Octave*: The octave of the notes to be played, the *Key*: The note used as the tonal center for the scale, and the *Frets*: The number of frets placed on the canvas. After the user selection, the preferred musical instrument setting can be saved for future usage as shown in the Figure 5.

When the musical system is set in the *Keyboard Mode*, a piano-like keyboard is drawn on the canvas at the center of the screen as shown in Figure 6. The keyboard layout has one octave and a total of 12 keys. Each individual key corresponds to one note in a typical Chromatic note scale. Based on the x coordinate of finger touch point on the 2-D canvas, the corresponding sound is generated by Csound API.

The mobile app draws a canvas with the frets/keys on it and uses the screen dimensions to generate ratios that are compared to values associated with musical notes in the activity. The value of these notes is further passed to Csound API to produce the corresponding sound frequency. Each musical note corresponds to a specific frequency range [27]. A typical note-frequency relationship is shown in Table 1.

Feedback Visualization

Another major feature of the proposed musical system is the visualization of system feedback. The finger activities of a user are displayed in a graphic format on the screen of mobile devices. The responsive animation on touch screen is currently supported by Android operating system's *show_touches* functionality. This finger touch indicator provides a low-latency response with ripple-like effects for all music playing events as shown in Figure 3 and in Figure 6. The effect has also been successfully deployed to both the main activity and settings activities.

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-300

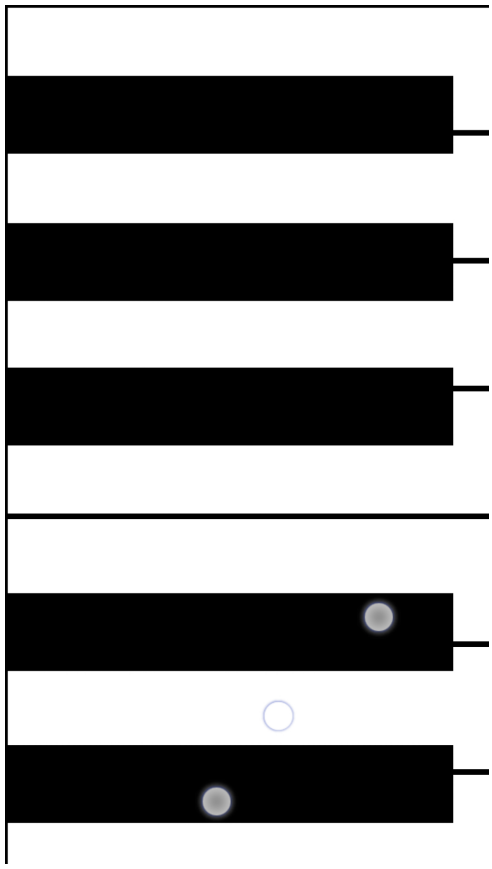

Figure 6. *The keyboard instrument is simulated by the proposed musical system. The ripple-like effect provides important system feedback for finger touch activities when a user plays music.*

The ripple-like effect makes use of a RippleView that draws a canvas on the screen of mobile devices. More specifically, a radius with a timed duration and framerate is created for the effect of an expanding circle rooted from the touch point. The ripple can be further customized to a specific color, color alpha, duration, and frame rate. It can be also stacked to create a double ripple effect. The simulating musical instruments hence provide more tactile system feedback with this type of touching effects.

Latency Reduction

Android's audio stack is not designed for real-time audio generation. The system performance is hence degraded shortly after the music playing starts. We introduce ProGuard [8] tool aiming to reduce the audio generation latency of the proposed musical system. ProGuard is integrated in the Android build system to help reduce and optimize code. This tool removes unused code and renames fields, classes, and methods with meaningless names to help make the application run faster. In addition, ProGuard could help users save more system space on mobile devices due to the smaller *.apk* file of mobile apps.

We took advantage of the ProGuard tool and build smaller and faster mobile applications. More specifically, we implement the features of ProGuard by modifying the default configurations in the *proguard.cfg* file. We modify the configuration file by specifying certain usage options provided by ProGuard. We mainly use the *Keep* options in our configuration file to preserve our classes that extend the different classes provided by Android. For example, we will preserve the extended android classes such as *android.app.Activity*, *android.app.Service*, *android.app.Application*, *android.app.backup.BackupAgentHelper*, *android.app.content.ContentProvider*, and a few others. These android classes are protected from being renamed or removed in order to keep the intended interactivity with the user, the backup and restore functionality, the environment for the application, the user interface, and the published data associated with our application. In addition, we preserve the names of the classes with native methods because we do not want these methods to be removed from our classes which are vital to our application. Moreover, we implement ProGuard usage options to make sure certain aspects of code are not skipped and to help improve the efficiency of the system code. For example, we do not want to ignore any non-public library classes and class members due to the fact that some of the system classes reside in the same packages as the library classes. Finally, we use the ProGuard optimization options to improve the quality of the system code. We specify ten optimization passes in order to improve the efficiency and the overall application. The size of the *.apk* file is hence reduced from 3740 kilobytes to 3362 kilobytes.

Results

Our mobile musical instrument simulation system is developed on Android platform using the Java language with the support of the API of Csound 6 [7], a venerable and open source audio synthesis system released under LGPL. More specifically, we divide the touch screen into numerous intervals where each interval represents a fret for string instruments or a key for keyboard instruments. When a user clicks the key, the corresponding frequency data of the musical note is transmitted to Csound API for sound generation. Multiple presses are allowed, and thus the creation of chords. The mobile app is able to generate various tones using different musical instruments.

We perform two rounds of experiments to validate the efficiency and reliability of the proposed musical system. We first

Table 1. The relationship between Chromatic scale notes and their corresponding audio frequency values.

Musical Notes	Audio Frequency
C	261.62 HZ
C#	277.18 HZ
D	293.67 HZ
D#	311.13 HZ
E	329.63 HZ
F	349.23 HZ
F#	369.99 HZ
G	392.00 HZ
G#	415.31 HZ
A	440.00 HZ
A#	466.16 HZ
B	493.88 HZ

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-300

quantitatively evaluate the sound reproduction latency of the musical system with individual tones. The experiments are conducted on one regular laptop computer. The machine has an Intel Core i7-3632QM CPU @ 2.20GHz processor with 8 GB memory and runs on Windows 10(x64) platform. We also qualitatively evaluate the performance of the proposed musical system in a survey with normal people. The survey is conducted with a Samsung Galaxy S6 edge smartphone with 64 GB memory and runs on Android 5 platform.

In the first round of experiments, the proposed musical system is quantitatively tested for the latency of sound generation. We first create a simple musical system to compute the frequency value for each Chromatic note and further generate the corresponding sound of that specific note using standard Android *AudioTrack* class. The simple musical system produces individual tone for each note of the Chromatic scale in the frequency range between 261.62 HZ and 493.883 HZ as shown in Table 1. We test each note for 1,000 times using a virtual device on Android Studio and report the average response time in milliseconds. As shown in Table 2, the average response time of our musical system is significantly shorter than that of standard Android audio stack. The experimental results show ProGuard tool can greatly reduce the system computation time and build smaller and faster mobile applications.

In the second round of experiments, we qualitatively test the performance of sound generation of the musical app, specifically the ability of Csound API to produce quality sound. A survey is conducted with the people unware of the existence of this type of musical applications. Most participants provide positive feedback to short musical scores played for them when examining the quality of sound produced by the mobile app. Some provided feedback ranged from "it sounds more realistic than many mobile devices" to "the produced sound was very clear." Receiving positive feedback bolsters the idea that the musical system produces quality sound that is well received by users.

Conclusions

With the increasing number of mobile devices and their characteristic of portability, more and more entertainment applications have been developed on mobile platforms. One promising direction for mobile app development is to simulate various musical instruments on mobile devices, providing users a real-time music playing experiences at any time and from anywhere. While some explorations have been conducted on iOS system, the performance of musical instrument simulation on Android system is largely unknown due to the high latency of audio reproduction with standard Android audio stack. In this work, we develop an effective mobile musical app on the Android platform for musical instrument simulation. The proposed musical system features two major functional components: (a). A convenient user interface for playing string and keyboard instruments on the touch screen supported by most mobile devices. (b). The visualization of system feedback for finger touch activities, which strengthens the tactile playing experience for users. We also optimize the musical system with ProGuard tool designed specifically for Android and with the API low-latency techniques. These technologies are proved to be effective for creating smaller and faster mobile apps.

Our mobile musical system has been shown to be convenient and effective in simulating various musical instruments. Some potential improvements are achievable for the proposed system in the near future: (a). Strengthen the physical response of the mobile system by introducing the vibrating functionality supported by most mobile devices to further improve the quality of playing experiences. (b). Extend the application over the web to enhance user social communication experiences in the ever-expanding world of social networking. (c). Automatically display the specific note whenever a user touches a certain fret on the canvas and hence allow the user to know the specific note he or she is playing. (d). Incorporate more musical instruments for simulation in the form of icon (picture) selection. (e). Promote user entertainment experience with scores computed based on the performance of music playing. The ultimate goal of the musical system is to demonstrate the feasibility and the potential for converting any mobile device into a particular musical instrument that can provide real-time music generation and interactive visual feedback.

Acknowledgements

This work was partially supported by the Frostburg State University President's Experiential Learning Enhancement Fund Program (PELEF) and by a FSU Foundation Opportunity Grant (# 30593).

References

[1] X. Pan, T. Cross, L. Xiao, and X. Hei, "Musical examination and generation of audio data," in *SPIE Symposium on Electronic Imaging (SPIE-EI)*, (San Francisco, CA), 2015.

[2] X. Pan and K. Free, "Interactive real-time media streaming with reliable communication," in *SPIE Symposium on Electronic Imaging (SPIE-EI)*, (San Francisco, CA), 2014.

[3] X. Pan and S. Lyu, "Region duplication detection using image feature matching," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 5, no. 4, pp. 857–867, 2010.

[4] X. Pan, X. Zhang, and S. Lyu, "Exposing image forgery with blind noise estimation," in *ACM Workshop on Multimedia and Security (MM&Sec)*, (Buffalo, NY), 2011.

[5] X. Pan, X. Zhang, and S. Lyu, "Detecting splicing in digital audios using local noise level estimation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (Kyoto, Japan), 2012.

[6] X. Pan, X. Zhang, and S. Lyu, "Exposing image splicing with inconsistent local noise variances," in *IEEE International Conference on Computational Photography (ICCP)*, (Seattle, WA), 2012.

[7] R. Boulanger, *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming*. MIT Press, 2000.

[8] E. Lafortune, "Proguard — optimizer and obfuscator in the android SDK." DroidCon, London, Oct. 2012.

[9] G. Wang, "Principles of visual design for computer music," in *International Computer Music Conference (ICMC)*, (Athens, Greece), 2014.

[10] G. Wang and P. R. Cook, "The audicle: a context-sensitive, on-the-fly audio programming environ/mentality," in *International Computer Music Conference (ICMC)*, (Miami, FL), 2004.

[11] G. Wang, R. Fiebrink, and P. R. Cook, "Combining analysis

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-300

Table 2. Comparison of the sound generation latency of the Android audio stack and that of the proposed musical system.

Chromatic Scale Notes	Latency for Android Audio Stack (in milliseconds)	Latency for Proposed Musical System (in milliseconds)
C	190.111	3.939
C#	192.490	4.144
D	270.225	2.624
D#	279.923	3.497
E	196.380	6.064
F	192.036	6.404
F#	204.950	5.400
G	187.897	2.868
G#	230.765	3.323
A	200.880	3.350
A#	223.824	5.109
B	192.385	4.455

and synthesis in the chuck programming language," in *International Computer Music Conference (ICMC)*, (Copenhagen, Denmark), 2007.

[12] A. Misra, G. Wang, and P. R. Cook, "sndtools: Real-time audio dsp and 3d visualization," in *International Computer Music Conference (ICMC)*, (Barcelona, Spain), 2005.

[13] J. Oh and G. Wang, "Converge: An omni- biographical composition," *Computer Music Journal Emile*, vol. 9, 2011.

[14] S. Smallwood, D. Trueman, P. R. Cook, and G. Wang, "Composing for laptop orchestra," *Computer Music Journal*, vol. 32, 2008.

[15] H. Choi and G. Wang, "Lush : An organic eco + music system," in *International Conference on New Interfaces for Musical Expression*, (Sydney, Australia), 2010.

[16] A. Tanaka, "Mobile music making," in *International Conference on New Interfaces for Musical Expression*, (Hamamatsu, Japan), 2004.

[17] G. Schiemer and M. Havryliv, "Pocket gamelan: Tuneable trajectories for flying sources in mandala 3 and mandala 4," in *International Conference on New Interfaces for Musical Expression*, (Paris, France), 2006.

[18] S. Strachan, P. Eslambolchilar, R. Murray-Smith, S. Hughes, and S. O'Modhrain, "Gpstunes: Controlling navigation via audio feedback," in *The 7th International Conference on Human Computer Interaction with Mobile Devices &Amp; Services*, (New York, NY), 2005.

[19] A. Tanaka, G. Valadon, and C. Berger, "Social mobile music navigation using the compass," in *International Mobile Music Workshop*, 2007.

[20] M. Rohs, G. Essl, and M. Roth, "Camus: live music performance using camera phones and visual grid tracking," in *International Conference on New Interfaces for Musical Expression*, (Paris, France), 2006.

[21] L. Gaye, L. E. Holmquist, F. Behrendt, and A. Tanaka, "Mobile music technology: Report on an emerging community," in *International Conference on New Interfaces for Musical Expression*, (Paris, France), 2006.

[22] A. Tanaka and P. Gemeinboeck, "A framework for spatial interaction in locative media," in *International Conference on New Interfaces for Musical Expression*, (Paris, France, France), 2006.

[23] G. Geiger, "Pda real time signal processing and sound generation on handheld devices," in *International Computer Music Conference (ICMC)*, (Singapore), 2003.

[24] G. Geiger, "Using the touch screen as a controller for portable computer music instruments," in *International Conference on New Interfaces for Musical Expression*, (Paris, France), 2006.

[25] G. Wang, "Ocarina: Designing the iphone's magic flute," *Computer Music Journal*, vol. 38, no. 2, pp. 8–21, 2014.

[26] G. Wang, J. Oh, and T. Lieber, "Designing for the ipad: Magic fiddle," in *International Conference on New Interfaces for Musical Expression*, (Oslo, Norway), 2011.

[27] O. Jorgensen, *Tuning: containing the perfection of eighteenth-century temperament, the lost art of nineteenth-century temperament, and the science of equal temperament, complete with instructions for aural and electronic tuning*. Michigan State University Press, 1991.

Author Biography

Xunyu Pan received the B.S. degree in Computer Science from Nanjing University, China, in 2000, and the M.S. degree in Artificial Intelligence from the University of Georgia in 2004. He received the Ph.D. degree in Computer Science from the State University of New York at Albany (SUNY Albany) in 2011. From 2000 to 2002, he was an instructor with Department of Computer Science and Technology, Nanjing University, China. In August 2012, he joined the faculty of Frostburg State University (FSU), Maryland, where he is currently an Assistant Professor of Computer Science and the Director of Laboratory for Multimedia Communications and Security. Dr. Pan is the recipient of 2011∼2012 SUNY Albany Distinguished Dissertation Award and three-time (2013, 2014, and 2015) FSU Summer Research Award. His publications span peer-reviewed conferences, journals, and book chapters in the research fields of multimedia security, image analysis, medical imaging, communication networks, computer vision and machine learning. He is a member of the ACM, IEEE, and SPIE. (Corresponding Author: xpan@frostburg.edu)

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-300

Jacob Wilson received the B.S. degree in Physics from West Virginia Wesleyan College in 2013. He is currently working toward the M.S. degree in Computer Science at Frostburg State University (FSU). He is now a Graduate Assistant in Department of Computer Science and Information Technologies at FSU.

Megan Balukoff received B.S. degree in Computer Science with Honor from Frostburg State University in 2015. She is currently working toward the M.S. degree in Computer Science at Frostburg State University (FSU). She is now a Graduate Assistant in Department of Computer Science and Information Technologies at FSU. She is also a member of Upsilon Pi Epsilon Computer Honor Society.

Anyi Liu is an Assistant Professor and the Associate Director of the Information Analytics and Visualization (IAV) Center at Indiana University-Purdue University Fort Wayne (IPFW). He has extensive experience in information security, intrusion detection and prevention, and digital forensics. He is a senior member of IEEE and has received his Ph.D. degree from George Mason University in 2012.

Wenjuan Xu received the Ph.D. degree in Information Technology from the University of North Carolina at Charlotte. She is currently an Assistant Professor of the Department of Computer Science and Information Technologies at Frostburg State University, Maryland.