Interactive Virtual Percussion Instruments on Mobile Devices

Zhimin Ren and Ming C. Lin*
University of North Carolina at Chapel Hill
http://www.cs.unc.edu/vMusic

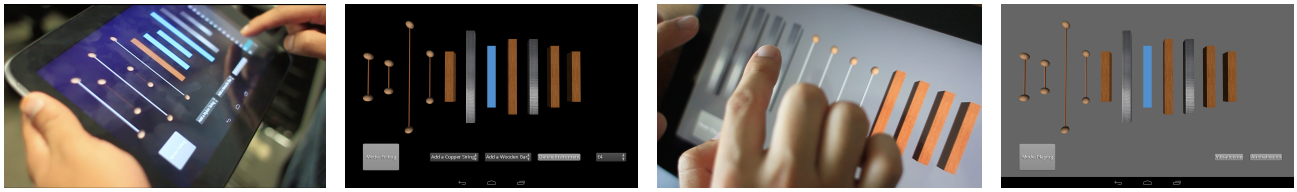

Figure 1: *The two images on the left show a user editing virtual musical instruments and a screenshot of the mobile application in* ***editing mode*** *to create a personal, reconfigurable instrument, while the two on the right show* ***playing mode*** *to play the customized instrument.*

Abstract

We present a multimodal virtual percussion instrument system on consumer mobile devices that allows users to design and configure customizable virtual percussion instruments and interact with them in real time. Users can create virtual instruments of different materials and shapes *interactively*, by editing and selecting the desired characteristics. Both the visual and auditory feedback are then computed on the fly to automatically correspond to the instrument properties and user interaction. We utilize efficient 3D input processing algorithms to approximate and represent real-time multi-touch input with key meta properties and adopt fast physical modeling to synthesize sounds. Despite the relatively limited computing resources on mobile devices, we are able to achieve rich and responsive multimodal feedback based on real-time user input. A pilot study is conducted to assess the effectiveness of the system.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Auditory (non-speech) feedback H.5.2 [Information Interfaces and Presentation]: User Interfaces—Prototyping H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing—Modeling H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing—Systems;

1 Introduction

Multi-touch mobile devices have brought a disruptive change to our daily interaction with computing devices. However, these mobile devices typically posses limited computing power due to their requirement for long battery life. Thus, mobile devices present an interesting yet challenging platform for developing VR and 3D interactive applications. In exploring the adoption of multimodal interaction on mobile devices, virtual music playing systems emerges among such applications due to its need to involve multiple senses and more complex interactions to master music playing.

A plethora of music related applications specifically made for mobile platforms can be found. Currently, almost all of them use only pre-recorded music or audio sample playbacks. As a result, the variation in user performance, such as where the instruments are hit and how hard they're struck, would result in no difference in the played music unlike the real-world experience. A few existing mobile apps allows multitouch interaction, but they often do not take advantage of the user's physical interaction with virtual instruments to modulate the sounds (in terms of volume and timbre) as in the real world. More complex user interactions, which can be easily captured by VR 3D user interfaces like motion trackers and digital gloves, are difficult to handle on consumer multi-touch screens. Moreover, due to limited computing power, current applications cannot provide realistic and dynamic synthetic sound effects, which are usually achieved by physical modeling and directly driven by the user motion.

In this paper, we present a reconfigurable virtual percussion instrument system on a consumer mobile device that is responsive to rich user interactions and capable of generating dynamically varying sound based on real-time user control. The main characteristics of our virtual instrument systems are:

- **A flexibly reconfigurable virtual percussion instrument system,** with which users can create their own percussion instruments by choosing from two common types of instruments (i.e. string vs. keyboard) and multiple materials, as well as mixing and editting the instruments based on pitches. This capability allows quick prototyping and personalization of virtual instruments.
- **An effective interaction processing algorithm** that handles strike and slide actions. It tracks in real time multiple finger inputs and scales to a large number of virtual objects. It also estimates the *contact forces* and *velocity* required for physics-based sound synthesis directly from different types of touch events.
- **A fast physical sound synthesis model** to generate sound effects that are dynamically varying and truly reflecting user's physical interaction with the device.

The type of optimizations we have made for the virtual musical instruments on mobile devices include (1) translating multitouches into physically meaningful input, i.e. contact forces and estimated velocity as 'impact' to activate physic sound synthesis computation; (2) the choice of physical models that are based on analytical formulation for strings and bars, thereby optimizing the computational model for real-time performance of physical sound synthesis

*e-mail: lin@cs.unc.edu

on a mobile device with limited compute resources; (3) extracted material properties from recorded audio that is transferrable to any geometry *at runtime*.

To the best of our knowledge, this system is the first virtual musical app on *mobile devices* that allows the users to simultaneously create multiple types of *personalized* instruments of different materials (metalic, wooden, plastic, etc.) based on *physical interaction* and allow the user to play and experiment with different virtual instrument configurations interactively, *without any pre-recording of the audio samples*.

2 Previous Work

Novel interfaces have been explored for virtual instruments [Chuchacz et al. 2007; Miranda and Wanderley 2006; Weinberg and Driscoll 2007]. However, none of them is as intuitive or easy-to-use for average users as multi-touch interfaces. Since the introduction of multi-touch screens on consumer smartphones and tablets, capacitive multi-touch hardware has become commonplace. On such devices, music applications like the *GarageBand* [Apple 2014] and *djay* [Algoriddim 2014] have become popular. Nonetheless, these applications all store static sound samples and play them at run time when triggered by user touches. Some of them are configurable, but they are all limited to simple selection of instruments and do not offer variations that match an instrument's physical properties like materials and shapes.

Recently sound synthesis has captured much attention in computer graphics. However, most of these physics-based algorithms do not run in real time on mobile devices, with a few exception of those using modal sound synthesis [Adrien 1991; Shabana 1997]. Unfortunately, this approach requires pre-computation called *modal analysis*, which renders it infeasible for interactive applications that alters object geometry *and* materials at run time. Acceleration techniques [Raghuvanshi and Lin 2006; Ren et al. 2010; Ren et al. 2012; Sterling and Lin 2015] all require siginificant pre-computation that enables the application to gracefully degrade audio quality that is simply not possible with virtual prototyping and real-time *dynamic alteration* of virtual instruments.

3 User Interface Design

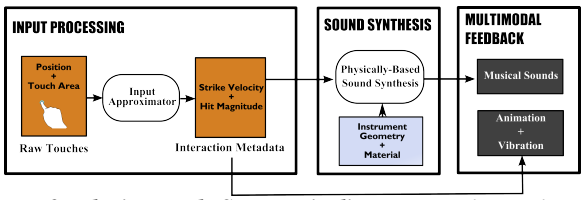

Figure 2: ***Playing Mode System Pipeline:*** *Raw multi-touch events registered by touch screen are processed by the* ***Input Approximator*** *and interpreted as meta interaction data. These data are used to drive the efficient physically-based sound synthesis module, which takes the instrument geometry and materials defined in* ***editting mode****. The interpreted interaction data also determine the dynamic animation and vibration the user experiences. Together richly varying multimodal feedback that corresponds to the user input is computed in real time.*

The mobile application we present provides two distinctive modes for these two different uses, namely *editing mode* and *playing mode*. On application startup, users are presented with the editing mode. A mode switching button is shown on the buttom left of the screen (see Figure 1). The button label tells users which mode they are in, and when tapped, the application is quickly toggled into the other mode. When users are editing instruments, they get the visual feedback for what the instruments' shape and materials are, but they might also want auditory feedback on what the instruments sound like. In this case, users can focus on the instruments, easily tap the mode switching button, enter into playing mode, hit the instrument for sounds, and finally tap back into editing mode and continue with editing without much cognititive overhead. To further facilitate easy mode, we assign different background colors for the two modes: BLACK for editing and GREY for playing. As users spend more time with this application, they naturally associate one mode with one background. Once that association is established, without even looking at the text on the button, users can smoothly distinguish and switch between the two modes, resulting in even smaller cognitive overhead.

3.1 Editing Mode

Editing mode is for users to create and configure their own customized virtual musical instruments to their liking. On application startup, an empty screen is presented and waiting for users' creation. Users are allowed to select from two types of instruments, namely *string* and *bar* or keyborad instruments. In addition, for each type, users can choose a material. Two dropdown menus at the bottom of the screen are shown for choosing the *type* and *material*. From the first menu, users are presented with the option to add a copper string or a nylon strying, while the second menu presents the ability to add a wooden bar, a metallic bar, and a plastic bar. When one of these menu items is tapped, a musical instrument of the corresponding type and material is created. Both the sound and the visual renderings of the instruments match the chosen type and material. Users can combine these instrument types and materials and create a variety of instrument configurations.

Moreover, users can edit any existing instrument on the screen. By touching an instrument, users express the intent to *select* it. When an instrument is selected, it is animated, and the device also vibrates to notify users of a successful selection. Users can perform two types of editing on the selected instrument, namely *pitch configuration* and *deletion*. A third dropdown menu lists musical notes from C_3 to B_4, a total of 14 pitches. Users can select any pitch from the list and assign it to the selected instruments. The length of the instrument also automatically changes corresponding to the chosen pitch. This realistically matches the physical properties of the virtual instruments' real-world counterparts.

3.2 Playing Mode

Playing mode is intentionally designed to be simple, so that users can focus on interacting with the instruments they created and configured. To provide a multimodal experience while still giving users control, we allow users to toggle the animation of instruments (visual feedback) and vibration of the device (haptic feedback) when an instrument is being played. Therefore, the user interface of playing mode only consistis of the instruments and two small buttons shown at the buttom right of the screen for toggling animation and vibration (see the screenshot of playing mode in Figure 1).

4 Algorithmic Design

To provide richly, detailed, and dynamic responses that are driven by user interaction in three-dimensional space and realistically match the geometries and materials of the musical instruments, real-time physical modeling is the ideal choice. Physical modeling realistically reflects real-world phenomena which are the most familiar to users and matches their expections. However, faithful

physical modeling requires intense computing power, which is usually far from what is currently available on mobile devices. Two categories of challenges are present in the playing mode. The first is how to capture and process complex user input, and the second is how to compute the rich multimodal feedback in real time. We present an algorithmic system that addresses both challenges. Figure 2 shows the pipeline of the system, in which the *input processing* module converts raw multi-touch inputs into interaction metadata that are used to drive the *sound synthesis* module, as well as animation and vibration as part of the *multimodal feedback*.

4.1 Input Processing

When users interact with musical instruments in real world, complex dynamics happen in three-dimensional (3D) space. However, a multi-touch screen can only capture interactions in 2D. To create a responsive and immersive experience, we need to close the gap between the two. Moreover, unlike traditional virtual applications on personal computers, where mouse and keyboard are usually the input devices, on multi-touch enabled devices, we are required to simultaneously track interaction of up to 10 fingers (i.e. 10 contact areas) with virtual objects, not to mention this all needs to happen with much more limited computing resources. Therefore, we are present with the challenge to effectively reconstruct, approximate, and represent 3D interaction in real time.

4.1.1 Reconstruct 3D Interaction

We reconstruct 3D interaction solely based on the limited 2D touchscreen events exposed by consumer tablets. Such touch events are usually registered as *touch begin*, *touch continue*, and *touch end*. Accompanied with these events are *touch position* (an obsolute position x pixels right and y pixels down from the upperleft corner of the screen) and *touch area* (the square area in pixels that approximates a finger touch area on the screen). With these limited 2D data, we derive 3D interaction data critical to generating sound effects as follows.

Position of the contact. Hitting an instruments at different positions generates different sound effects. This data is retrieved by shooting a ray from the camera position in the virtual 3D world to the touch position mapped on the viewport and computing the intersection between this ray and virtual instruments. The intersection point is used as the contact position.

Type of the interaction. An interaction can be either a *strike* or a *slide*. A *strike* has a single contact position, while a *slide* is a continuous contact. They have different computation complexities. If we treat a slide the same as a strike, we would be constantly tracking and computing intersections with virtual objects, which is expensive. In addition, the problem is exacerbated as we support ten fingers interacting with a large number of virtual objects.

Based on the *touch begin* event, we distinguish between the two types of interactions. On a touch begin event, if the touch position is intersecting with any virtual object, we identify it as a strike interaction. If the device sees a *touch continue* event, it treats the interaction as a sliding event. For a sliding event, the intersection is computed very efficiently with the *input approximation* described in Section 4.1.2.

Force of the contact. The maginitude of the force is an important variable, because a light touch should only induce a low volume musical tone, while a forceful strike or slide action should produce a loud sound. However, multi-touch screens on consumer mobile devices usually do not have capabilities to capture force or pressure. For *strike interaction*, we use the *touch area* returned by touch screens to emulate the magnitude of the applied force. For a *slide action*, we use the *velocity* derived from the traveled distance of a finger and scale the magnitude of force proportionally. The contact forces along with the estimated velocity (see next) is required to compute synthesized sound effects.

4.1.2 Input Approximation

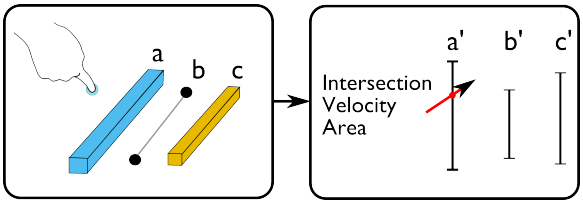

Figure 3: ***Input Approximation:*** *Dimentionality reduction that abstracts and represents a 3D space configuration with a 2D one.*

As alluded in Section 4.1.1, faithfully computing continuous intersections between touch events from ten fingers and all the virtual objects is expensive. Moreover, this computation has to happen in real time (30 fps) with only a small percentage of the available processing resources. In this case, directly computing these intersections in 3D is infeasible, because the user experience is unacceptable due to lag and unresponsiveness.

We propose an input approximation that reduces dimentionality and abstracts and represents the 3D interactions effectively with a 2D configuration. Figure 3 illustrates an example of such approximation process. On the left, a user is interacting with the multi-touch screen, which displays the virtual instruments, namely Bar a, String b, and Bar c. Without any approximation, in each frame, we would be casting a ray from the camera position to the touch point in the viewport and then compute the intersection between the ray with all three objects in 3D space. On the right, we present the approximation which speeds up this intersection computation. We first compute the bounding box of a virtual instrument and then represent the original instrument geometry with a line segment that spans the longest dimension of its bounding box. As shown in the image on the right, Bar a, String b, and Bar c are respectively approximated by Line a', b', and c'. When a *touch continue* event happens, we identify it as a sliding action as described in Section 4.1.1 and compute an *interaction line segment*, which starts at the projected touch position in last time step and ends at the projected touch position in the current time step (shown as the red arrow in Figure 3, where the arrow indicates the direction of this interaction). Once we represent 3D instruments and touch events as line segments, we compute inteserction between the instrument and interaction line segments, and the intersections in 3D space are approximated as contact locations in the 2D configuration (shown as the red dot in the right image in Figure 3). Based on the sliding velocity and contact area, we also scale the applied force appropriately. The described input approximation significantly reduces computation complexity and guarantees performant response computation that is critical to real-time applications.

4.2 Sound Synthesis

In order to build a virtual musical instrument system that allows the users to interactively create any configuration and choose any material, as well as automatically and correctly responding to users' dynamic and rich interaction, we need to compute sound samples in real time with physical modeling. Existing, known mobile music applications in general play recorded or pre-computed sounds at run-time.

We adopt an analytical formulation for bars and strings, similar to the physical waveguide synthesis methods [Cook 2007]. Specifi-

Table 1: *Are responses as expected? Scale: 0 (No) to 10 (Yes)*

String: sound variation with different hit points	Bar: sound variation with different hit points	String: shape change when changing pitch	Bar: shape change when changing pitch	String: sound variation with different strike velocities	Bar: sound variation with different strike velocities
7.33 ± 1.86	8.00 ± 0.89	9.17 ± 0.75	9.17 ± 0.75	6.50 ± 3.02	5.50 ± 3.02

cally, string instruments are modeled with the bowed string physical model [Jaffe and Smith 1995]. For bar instruments, a banded model for bowed bars [Essl and Cook 1999] is used. When a strike happens, one single impulse using the approximated velocity and contact forces is exerted to the sound synthesis module. For a slide, a series of impulses are applied corresponding to a series of contact points.

Material Properties: The user can choose and edit the material properties that are automatically extracted from the real audio recordings using [Ren et al. 2013a]. This feature allows the user to experiment with different configurations of musical instruments and examine the sound effects due to variation of materials interactively.

4.3 Implementation Details

The virtual musical instrument system described in this paper is implemented on a consumer mobile device, the Google Nexus 10 tablet with a Dual-core A15 mobile CPU and 2GB of RAM. Android 4.3 Jelly Bean [Google 2013] is the mobile operating system. To render sound, we use the audio library in Android SDK and render audio at 44,100 Hz. Rendering and animation are run at 30 fps.

5 Pilot Study

We performed a limited pilot study with six subjects: five male, one female, and age between 28 and 34. All results presented in this section are averaged over the six subjects with the standard deviation appended. At the start the study, we briefly demonstrate the functionality of the mobile application and present subjects with a questionnaire. Subjects can answer the questions at any time in the duration of the study. We observed all subjects quickly learned how to use the application immediately and started reconfiguring and playing the virtual musical instruments. Specifically, we asked the subjects to evaluate how easy the process was, and the result is shown in Table 2, and subjects generally considered it easy to operate the application.

Table 2: *Is it easy to do the following: 0 (Difficult) to 10 (Easy)*

Easy to generate sounds?	Easy to pick the desired instrument?	Easy to modify the chosen instrument?
9.17 ± 0.98	7.67 ± 2.25	8.33 ± 2.25

We evaluate how realisitc and natural the generated responses are by asking our subjects to score if the sound and geometry change match what they expect when they vary their interaction with the instruments. These questions and results are listed in Table 1. Sound variation with both different hit points and shape change, when changing pitch, is rated high. When subjects considered the sound variation with different strike velocities as expected, the score is lower. We suspect, due to the small-screen real estate on mobile devices, it is difficult for users to vary strike velocities.

We also measured the quality of the synthesized sounds. We asked if the generated sounds were realistic. 0 means 'no', and 10 means 'yes'. For string instruments, subjects responded with 8.50 ± 0.84, and for bar instruments, 7.50 ± 1.38. In addition, our subjects observed low latency when using the application. When asked "can you observe any latency" (0 means obsolutely yes, and 10 means obsolutely no latency), our subjects rated this 7.38 ± 2.93. Last but not least, we evaluate the multimodal experience. Our subjects are able to toggle on and off the visual feedback (animation of the instruments) and the haptic feedback (vibration of the device). We asked subjects which mode of experience do they prefer. Four out of the six subjects prefer the experience with only auditory and visual feedback, one likes all modals turned on, and one likes only auditory feedback on. We further asked the four subjects why they preferred the haptic feedback off, and they all responded that the device's vibration is too strong and generates a buzzing sound that interferes with the auditory feedback.

6 Conclusion and Future Work

We present a real-time physically based virtual musical instrument system that is reconfigurable and reallistically responds to user interaction. With efficient input handling and sound synthesis algorithms, we achieved this on a consumer mobile device with limited computing resources. In the future, we hope to add fully-featured and more complex shape editing, so that users can scale and sculpt virtual sounding objects on the fly. Providing user with an intuitive way of editing material can also be explored. A full study and analysis on the variation in user performance due to interaction with virtual percussion instruments, similar to [Ren et al. 2012; Ren et al. 2013b; Young and Serafin 2003] for string instruments, would be an interesting future direction.

Acknowledgment

This work is supported in part by National Science Foundation. We thank the reviewers for suggestions.

References

ADRIEN, J.-M. 1991. Representations of musical signals. MIT Press, Cambridge, MA, USA, ch. The missing link: modal synthesis, 269–298.

ALGORIDDIM, 2014. djay for ipad. `http://www.algoriddim.com/djay-ipad`.

APPLE, 2014. Garageband for ios. `http://www.apple.com/ios/garageband/`.

CHUCHACZ, K., O'MODHRAIN, S., AND WOODS, R. 2007. Physical models and musical controllers: designing a novel electronic percussion instrument. In *NIME '07: Proceedings of the 7th international conference on New interfaces for musical expression*, ACM, New York, NY, USA, 37–40.

COOK, P. R. 2007. Real sound synthesis for interactive applications.

ESSL, G., AND COOK, P. R. 1999. Banded waveguides: Towards physical modeling of bowed bar percussion instruments. In *Proceedings of the International Computer Music Conference (ICMC)*, 321–324.

GOOGLE, 2013. Android 4.3 jelly bean operating system. `http://www.android.com/about/jelly-bean/`.

JAFFE, D. A., AND SMITH, J. O. 1995. Performance expression in commuted waveguide synthesis of bowed strings. In *In ICMC*, 343–346.

MIRANDA, E., AND WANDERLEY, M. 2006. *New digital musical instruments: control and interaction beyond the keyboard*. AR Editions, Inc.

RAGHUVANSHI, N., AND LIN, M. 2006. Symphony: Real-time physically-based sound synthesis. In *Proceedings of Symposium on Interactive 3D Graphics and Games*.

REN, Z., YEH, H., AND LIN, M. 2010. Synthesizing contact sounds between textured models. In *Virtual Reality Conference (VR), 2010 IEEE*, 139 –146.

REN, Z., MEHRA, R., COPOSKY, J., AND LIN, M. C. 2012. Tabletop ensemble: touch-enabled virtual percussion instruments. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 7–14.

REN, Z., YEH, H., AND LIN, M. C. 2013. Example-guided physically-based modal sound synthesis. *ACM Trans. on Graphics 32*, 1 (January), Article No. 1.

REN, Z., YEH, H., KLATZKY, R., AND LIN, M. C. 2013. Auditory Perception of Geometry-Invariant Material Properties. *IEEE Transactions on Visualization and Computer Graphics 19*, 4, 557–566.

SHABANA, A. 1997. *Vibration of discrete and continuous systems*. Springer Verlag.

STERLING, A., AND LIN, M. 2015. Integrated multimodal interaction using normal maps. In *Proceedings of Graphics Interface*, 33–40.

WEINBERG, G., AND DRISCOLL, S. 2007. The interactive robotic percussionist: new developments in form, mechanics, perception and interaction design. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, ACM, New York, NY, USA, HRI '07, 97–104.

YOUNG, D., AND SERAFIN, S. 2003. Playability evaluation of a virtual bowed string instrument. *Proc. of Conference on New Interfaces for Musical Expression*, 104 – 108.