Guitar Simulator: An Audio-Haptic Instrument for Android Smartphones.

Ben Cahill and Stefania Serafin
Aalborg University Copenhagen
A.C. Meyers Vænge 15, 2450
Copenhagen
Denmark

Bcahil09@student.aau.dk,
sts@create.aau.dk

ABSTRACT
The haptic capabilities of a smartphone are used in a model of plucked strings. The current implementation is simple yet shows promise for this type of interaction on the Android platform.

Keywords
Haptics, musical instrument, smartphone, guitar

1. INTRODUCTION
The smartphone platform offers a unique test bed for music-based experiments; high quality graphics, sound and haptic capabilities are built into most modern examples.

Haptic feedback has been shown to enhance a virtual musical experience [1][2] by increasing the realism of the experience via multisensory stimulation. Even relatively low fidelity stimulation across multiple sensory channels is capable of providing a convincing impression of an event.

The Android interface encourages tapping and stroking and many instrument simulations have been developed that exploit these actions [3][4]. A limitation in these simulations is the latency due to audio drivers that are not optimised for real-time performance (unlike Apples iOS audio system). This makes percussion simulations in particular unconvincing and difficult to use with any degree of accuracy.

Very few virtual instruments use active haptic feedback, although some use haptics as navigational cues in menus. The smartphone actuators are usually small eccentric motors that have a low maximum amplitude but have short rise and fall times.

These capabilities are employed in a simulation that works in three modalities; the visual, audio and haptic.

2. DESIGN

2.1 Platform
The hardware platform is comprised of a Samsung Galaxy S2 Smartphone and a Dell Studio XPS laptop PC. The user interaction, graphics and haptics are programmed in Processing, a Java-like high level IDE, and compiled for Android; the audio is programmed in PureData and runs on the Windows PC. The Smartphone is running the Processing sketch and sending OSC data to the PureData patch over a local network.

2.2 Guitar Simulator
The Guitar Simulator (GS) is an interactive application that provides visual, audio and haptic feedback in a virtual acoustic guitar. The user can pluck or strum the strings and adjust the string dampening and the plucking mass. A metronome triggers a change to a random diatonic chord at a given subdivision.

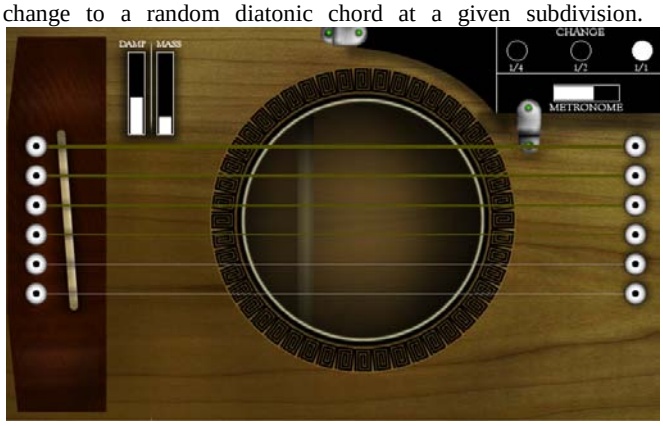

Figure 1. Guitar Simulator GUI

The user plays the strings by stoking across them. Additional GUI features are operated by tapping and dragging.

The strings are modelled graphically using a mass-spring-dampener model (MSD). The user's striking speed is sampled, the acceleration derived and multiplied by the mass to get the input force to the model. The centre mass is displaced at that initial time according to the MSD formula and decay over time accordingly. The string is drawn by interpolating between offset MSD models. The frame rate is set to 60fps.

The audio data is generated using a Karplus-Strong algorithm [5] in PureData. A short noise burst is fed into a filtered, attenuated delay line and this emulates the decay characteristic of the string. Changing the delay time changes the pitch of the tone. The input force is mapped to both the amplitude of the sound and to the filter in the delay line. This provides a duller, quieter tone at soft plucking levels and a louder, brighter tone at harder plucking levels.

Each string is tuned to its real world equivalent in standard tuning (E,A,D,G,B,E) and the corresponding MIDI note is sent to the string simulation.Each of the 7 diatonic chords is represented by an array of offsets per string which renders the chord voicing. The chords are based on the I chord having the same form as a first-position E major chord. When the metronome sends the change

message, a new chord from the diatonic array is selected. The key can be changed by adding an overall offset (-5 to +6 for all 12 keys) to the MIDI note number. Implementing rules to define key changes in this system would be relatively and allow the possibility of semi-automated composition.

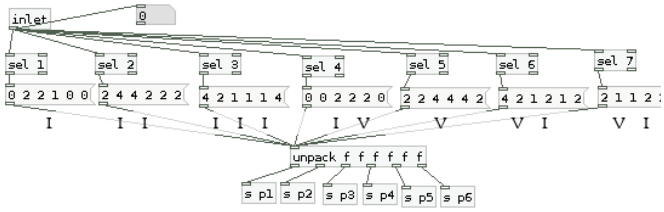

Figure 2. Chord selection Pd subpatch

The Android SDK allows the Smartphone vibrator to be turned on or off once per millisecond, giving a maximum vibration frequency of 500Hz. The vibration is relatively light but the position of the vibrators on the back of the phone provides a good haptic sensation to the fingertips. The amplitude of the vibration is fixed but can be manipulated using primitive Pulse Width Modulation. The primitive nature of the PWM is that its resolution is frequency dependant. For example, at 500HZ, no PWM is possible; at 250Hz only 3 PWM variations are possible.

The haptic feedback is therefore quite simple; a 10ms "on" signal is sent to the vibrator at the event of a string pluck. As the GUI and the input and haptic systems are all running on the Smartphone, there is no noticeable latency in the visual-haptic domain.

2.3 Conclusion

The Guitar Simulator has a lot of potential as a haptic instrument. Although not formally evaluated yet, informal tests with several Android users showed that even the simple level of haptic interaction in the current iteration provides an enjoyable sensation, congruent with the visual and audio aspects.

A possible venue for formal experimentation could be to assess the role of haptic feedback in enhancing realism and quality of the simulation.

The current system can serve as a platform for both exploring haptic feedback in hand-held musical instrument simulations and for computer assisted composition.

The next iteration of this work is to rewrite the audio code for an Android compatible system and integrate it into the application. Using the Karplus-Strong algorithm also facilitates the ability to render more detailed feedback. The interaction between the finger and an already vibrating string would add to the realism of the simulation and allow the user to dampen strings and play more expressively. The random chord change is not musically useful. A system to create dynamic chord progressions would also add interest for the player and the GUI could be updated to show the last 4 chords so a player may remember a favoured progression.

One interesting avenue of research could be to add decaying vibrations to the string, summing these then using PWM to create the composite decay in the haptic domain. As it stands, the limitations of the data permitted to be sent to the vibrators provide so simple way to execute composite events.

The current multi-touch implementation in Processing for Android is at the beta stage and somewhat unreliable. Allowing multi-touch interactions would provide a more complex and interesting user experience, as well as a closer analogue to the real-world input method.

3. REFERENCES

[1] Baillie , Lynne, David Beattie and Lee Morton. 2011. Feel What You Hear: Haptic Feedback as an Accompaniment to Mobile Music Playback. In *Proc. of IWS '11, August 30, 2011, Stockholm, Sweden.*

[2] Gunther ,Eric and Sile O'Modhrain. 2002. Cutaneous Grooves: Composing for the Sense of Touch. In *Journal of New Music Research 2002, Vol. 31, No. 1*

[3] Gillian, Nicholas, Sile O'Modhrain, Georg Essl. 2009. Scratch-Off: A gesture based mobile music game with tactile feedback. *In Proc. of NIME '09 Pittsburgh, Pennsylvania, USA.*

[4] Smule Ocarina. http://ocarina.smule.com/

[5] Karplus, Kevin and Strong, Alex. Digital Synthesis of Plucked Strings and Drum Timbres. Computer Music Journal, vol. 7, n.2, 1983.