SMARTPHONE AS CONDUCTOR'S BATON FOR VIRTUAL CONCERTS

Rasika Ranaweera, Michael Cohen, & Shun Endo
Spatial Media Group, U. of Aizu
Aizu-Wakamatsu, Fukushima-ken 965-8580
Japan
{d8121104, mcohen, s1160037}@u-aizu.ac.jp

ABSTRACT

We describe how smartphones can be used as a simplified baton of a ensemble conductor in virtual concerts. Alice, a 3D programming environment, can be used to develop interactive virtual spaces. One can drag-and-drop 3D objects and transform/arrange them in a WYSIWYG editor. Alice also provides a rich API to control and display objects, including some audio-related functions. We have created a virtual concert application in which instruments are arranged around a conductor (in this case the user) located at their center. A workstation running Alice can render such scenes, including displaying sound via stereo speakers. A user-conductor with a smartphone can simply point at a preferred instrument and tap to select or start playing. By tilting the device or sliding a control on the screen, volume of a selection can be adjusted. We used our own collaborative virtual environment (CVE) session server as the communication link between the smartphone or tablet and Alice. A middleware program reads orientation of the smartphone, sending such events to the CVE server. Another program, the Alice–CVE Bridge, retrieves these events through the server and selects/plays/adjusts instruments in the Alice virtual environment. Such explorations suggest the power of emerging mobile devices as generalized remote controls for interactive multimedia and ubiquitous computing.

1. INTRODUCTION

In an orchestra, a conductor is responsible for performing the pieces to be played. Using understandable baton movements, the conductor signals to the orchestra about tempos and tempo changes, and also cues the performers for their entrances. Gathering musicians and conducting an orchestra can be beyond the commitment of a general user who only has a taste for music and intention of live performance. With the emergence of virtual environments such as Alice, even general computer users can drag-and-drop 3D objects (cities, buildings, furniture, instruments, controls, animals and avatars) from a built-in gallery and arrange them to create attractive cyberworlds, or in this case a virtual concert with a few instruments. However, clicking on buttons or pressing keys to play instruments can be less appealing. It would be more realistic if the user could point and control the virtual concert like a conductor, sitting away from the monitor or screen (as shown in Figure 1). A modern smartphone (or a tablet) can be effectively used as a simplified baton instead of legacy pointing devices. By pointing at an ensemble, the user conductor can select an instrument, tap the "baton" screen to start/pause playing, and slide a control to adjust instrumental volume.

1.1. Smartphones

Contemporary smartphones not only have high–resolution multi–touch screens, high speed multicore processors, huge memory, Wi–Fi, and mobile broadband access, but they also run third-party applications using advanced application programming interfaces (APIs). A wide variety of programming languages— including Java (Google Android, BlackBerry), Python (Symbian S60), C/C++ (BlackBerry), and Objective C (Apple iOS)— is available for developing new applications, called "apps." Vendor companies encourage developers to create apps by opening marketplaces (such as Apple App Store, Blackberry App World, Google Android Market, etc.), providing integrated development environments (IDEs) (such as Eclipse, Xcode, BlackBerry JDE, NetBeans) or plug-ins (BlackBerry Java Plug-in for Eclipse), simulators, and code resources.

1.1.1. Gyroscope, accelerometer, and magnetometer

Besides previously mentioned features, smartphones also have variety of sensors (proximity, ambient light, and gyroscopic sensors). A gyroscopic sensor detects 3–axis angular acceleration around the X, Y and Z axes, enabling precise calculation of roll, pitch, and yaw. The gyroscope may complement an accelerometer, a sensor that detects a device's shake, vibration shock, or fall by detecting linear acceleration along one of three axes. Combined data from accelerometer and gyroscope enable a device to recognize approximately how far, fast, and in which direction it has moved in space. A magnetometer senses the Earth's magnetic field to determine its current orientation [1]. The most basic function of a magnetometer is to act as a compass, or to determine the direction that a user is facing. In order to minimize errors in sensors' reading, gyroscope, accelerometer, and magnetometer are used in conjunction.

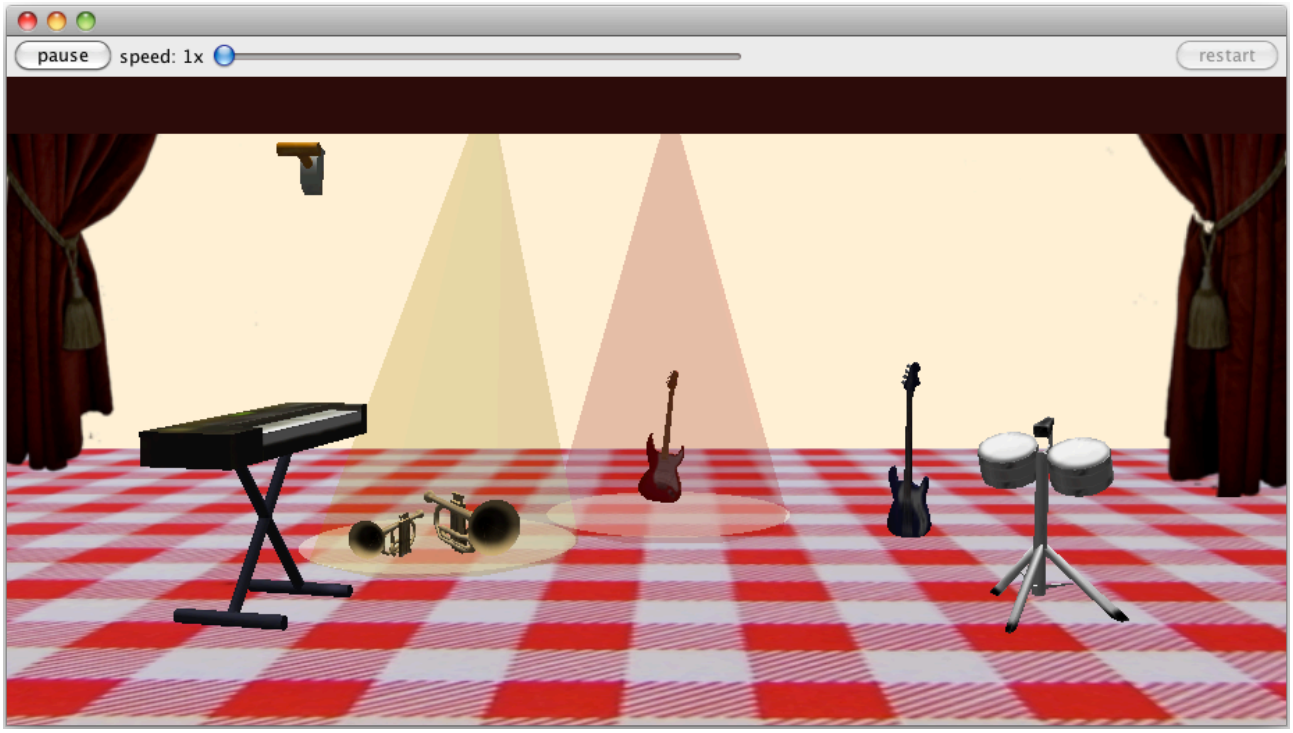


Figure 1. Virtual concert: Instruments are placed in a semi-circular arrangement to make it easy for a user-conductor to select among them. A selected instrument is graphically emphasized using a spotlight-like effect.

1.2. Alice

Alice [1] is an innovative object-oriented programming system from Carnegie Mellon University. It uses 3D graphics and a drag-and-drop interface for object manipulation and programming to develop interactive virtual spaces [2, 3, 4]. Alice 3 (Beta) provides a rich API to control and display objects, including some audio-related functions. Using a plugin for NetBeans, [2] Alice programs (scenarios) can be edited as Java outside the Alice native IDE. This allows invoking both the Alice native API and entirely different libraries.

2. OVERALL SYSTEM ARCHITECTURE

We have created a virtual concert scenario using Alice in which instruments are arranged with a conductor (in this case the user) located at their center. A workstation running Alice can render such scenes, including displaying auditory cues via stereo speakers. As previously mentioned, once programmed though a compatible language, spatial information of a smartphone can be accessed using the platform's API and transmitted via Wi–Fi. Depending on control device, a middleware program captures such events and sends to a collaborative virtual environment (CVE) session server. The extended Alice program, which also subscribes to the CVE server channels, receives events and trigger functions to select instruments in the virtual concert (as shown in Figure 2). Depending on the event, sound files associated with each instrument render sound using Java Sound API. So a user-conductor with a smartphone can simply point at a preferred instrument and tap to select or start playing.

[1] —www.alice.org—
[2] —www.netbeans.org—

2.1. Control soundscape using Java Sound API

As native Alice audio capabilities are limited to playing an audio file and adjusting volume before playing, the Java Sound API, which provides more complete control for effecting and controlling sampled audio and MIDI, was used to implement audio-related functions. In Java, a `Line` is an element of the digital audio pipeline, such as a mixer. Lines often have a set of controls, such as gain and pan, that affect the audio signal passing through the line. By accessing gain (`FloatControl.Type.MASTER_GAIN`) and panning (`FloatControl.Type.PAN`) controls of a line, dynamics and stereo mix of musical ensemble can be adjusted during audition.

2.2. CVE Client–Server Architecture

The CVE is a Java client–server protocol developed by the Spatial Media Group at the University of Aizu. Clients connect to a session server host via sharable channels [5]. When clients need to communicate with each other they subscribe to shared channels. The CVE server captures all updates (x, y, & z translation coordinates and $roll$, $pitch$, & yaw rotation coordinates) from session clients and multicasts (actually replicated unicast) to other clients via relevant channels. An extensibility "extra parameter" can also used to exchange information other than position details. This architecture allows multimodal displays to collaborate with each other, including mobile applications, motion platforms, map interfaces, spatial sound, and stereographic displays.

2.3. Alice–CVE Bridge

The Alice–CVE Bridge is a middleware program written in Java. Using a NetBeans plugin, an Alice scenario can be extended to connect to a CVE server. At runtime a connection is created and used to subscribe to a particular channel to which other clients are also subscribed. When position updates are received from the CVE server, rotation coordinate yaw can be used to determine which instrument is selected. Depending on the value of the extra parameter, an instrument can be started, paused, or have its volume adjusted.

2.4. Android–CVE and iOS–CVE Bridges

Both Android and iOS provide software development kits (SDKs) to access spatial information of respective devices [6]. Programming Android devices to connect to a CVE server is straightforward as the native languages are both Java. Once the package is deployed to the device, spatial information (devices's heading tracked using built-in magnetometers [6]) can be sent directly to the CVE server after the application launches. As iOS native programming is Objective C, a Java program can not be written to directly read such a device's spatial information. Instead such information is streamed to a TCP socket and a Java program can read the socket and relay such information to the CVE server, as seen in Figure 2. Even though this architecture is not as simple as the Android case, an end user need not be concerned with differences between platforms, except for the minor differences between the user interfaces (as shown in Figure 3).

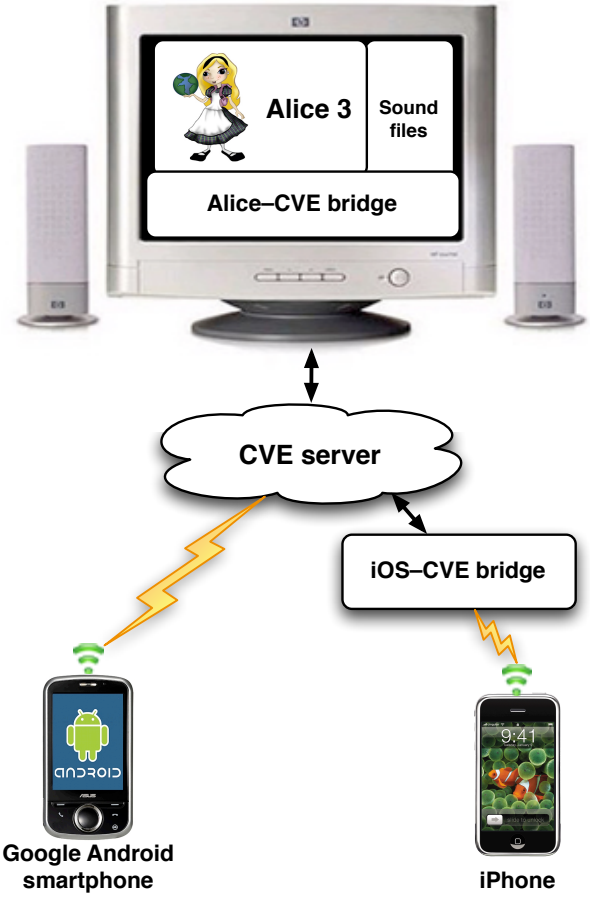
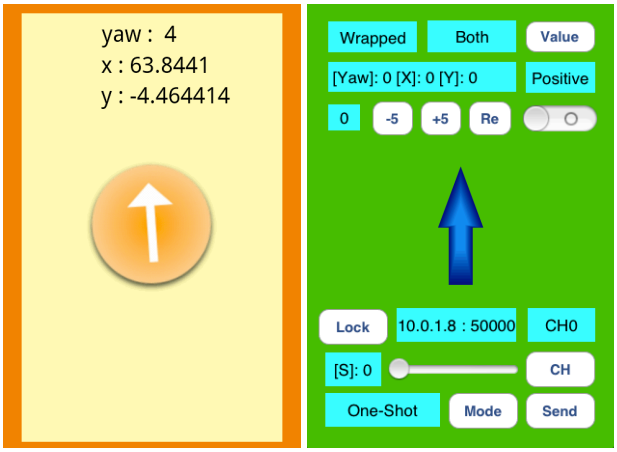


Figure 2. Alice scenarios can be extended to connect to a CVE session server, the communication link between the smartphones or tablets and Alice. Spatial information of a smartphone can be accessed using the device's API, and sent to distributed "cloud-based" services via Wi–Fi.

3. REALTIME PERFORMANCE

A workstation running Alice 3 renders the virtual concert scene, and stereo speakers attached to the workstation display the concert soundscape. A CVE server running on the same or a different computer accepts connections from Alice and smartphones. Once connections are established and matched and the phone is calibrated with the monitor setup, one can perform as a conductor in a virtual concert. When an instrument is selected by simply pointing at it, the base of the instrument is illuminated. One can tap on the screen to play/pause an instrument. When selected, an instrument is jiggled or its components dilated and contracted, and a spotlight appears until the instrument is muted, providing the conductor and audience with visual cues about the ensemble. One can also adjust volume/panning of a selected instrument by sliding corresponding controls on the smartphone screen.



(a) Android interface (b) iOS interface

Figure 3. An app which sends device's orientation to the CVE server, can be extended and used as a virtual baton interfaces on various smartphones.

3.1. Discussion

As previously mentioned, the CVE server may have various clients subscribed to the same channel as the virtual concert. When such clients update the server with rotation parameter yaw, which is used for instrument selection, it can mislead a user about current selection. As the operations are carried out according to the commands being sent via 'extra parameters,' the application is robust in other cases. Calibration of the phone before playing any instrument is required, and the user may perform it intuitively by adjusting a threshold. As the arrangement is

semi-circular, each instrument is assigned $\frac{\pi}{|\text{instruments}|}$ fraction.

Any angle within the range selects an instrument. Instead of sending continuous orientation events to CVE server, the app can be configured to respect an angular threshold. In this case events are sent only when the difference between current and previous orientation values exceed the threshold. Such a simple technique can be useful for the performance of the entire system. In Alice, motion of an object in space can be influenced by mass and inertia, making the motion more realistic. However such effects add delay. Even without intentional delays, there are unavoidable network delays between a client & CVE server, and a CVE server & Alice application. Substantial delays may not be preferred and can not be ignored especially for selections. In order to provide better control of using the system, some of the realistic effects of Alice had to be neglected.

4. CONCLUSION & FUTURE WORK

We have described how smartphones can be used as a simplified batons in virtual concerts. A user–conductor can point at an instrument to select it, tap the screen to start/pause playing, and slide a control to adjust volume. In an orchestra, a conductor signals to the performers using understandable baton movements. Even though our current approach does not have capabilities for such complex movements, smartphones provide sufficient information to capture device movements in 3D space. We have plans to improve our system with such complex but appealing capabilities and also implement spatialized sound system using HRTFs. The current implementation does not support dynamic addition/deletion/movements of instruments and selection of audio, but we hope to implement such features in the future. Even though using smartphones as pointing device [7] is not a novel idea, our experiments with controlling virtual concerts using smartphones suggest the power of emerging mobile devices as generalized remote controls for interactive multimedia and ubiquitous computing.

5. SPECIAL THANKS & ACKNOWLEDGEMENTS

We thank Shun Endo (Apple iOS development) and Hayato Ito (Google Android development) for their support in developing “Twinspin” apps.

6. REFERENCES

zw

[1] D. D. Rowlands and Daniel A James. Enhancing smart device applications using motion and location. In *CreateWorld: Working on the Edge: Creativity, Technology, and Innovation*, pages 36–46, Brisbane, Australia, November 2010.

[2] Matthew Conway, Steve Audia, Tommy Burnette, Dennis Cosgrove, Kevin Christiansen, Rob Deline, Jim Durbin, Rich Gossweiler, Shuichi Koga, Chris Long, Beth Mallory, Steve Miale, Kristen Monkaitis, James Patten, Jeff Pierce, Joe Shochet, David Staack, Brian Stearns, Richard Stoakley, Chris Sturgill, John Viega, Jeff White, George Williams, and Randy Pausch. Alice: Lessons learned from building a 3d system for novices. In *CHI: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 486–493, 2000.

[3] Stephen Cooper and Wanda Dann. Teaching objects-first in introductory computer science. In *SIGCSE: Proc. of the 34th SIGCSE Technical Symp. on Computer Science Education*, pages 191–195, 2003.

[4] Randy Pausch, Tommy Burnette, A.C. Capehart, Matthew Conway, Dennis Cosgrove, Rob DeLine, Jim Durbin, Rich Cossweiler, Shuichi Koga, and JeffWhite. Alice: Rapid prototyping for virtual reality. *IEEE Computer Graphics and Applications*, 15:8–11, 1995.

[5] Rasika Ranaweera, Nick Nagel, and Michael Cohen. Wonderland–CVE bridge. In *Proc. HC-2009:* 12^{th} *Int. Conf. on* **H***umans and* **C***omputers*, pages 91–98, Hamamatsu, Japan, December 2009.

[6] Michael Cohen, Rasika Ranaweera, Hayato Ito, Shun Endo, Sascha Holesch, and Julián Villegas. Whirling interfaces: Smartphones & tablets as spinnable affordances. In *ICAT:* 21^{st} *Int. Conf. on Artificial Reality and Telexistence*, Osaka, Japan, November 2011.

[7] T. Horiuchi, S. Takayama, and T. Kato. A pointing system based on acoustic position estimation and gravity sensing. In *3DUI: IEE Symp. on 3D User Interfaces*, pages 105–106, March 2011.

7. AUTHORS’ PROFILES

Rasika Ranaweera

Doctoral student at University of Aizu, Japan, whose research interests include multimodal interfaces, immersive virtual environments, and spatial sound.

Michael Cohen

Professor in the Dept. of Computer and Information Systems, University of Aizu, Japan, and Director of Spatial Media Group.