Some Principles of Visual Design for Computer Music

GE WANG

ABSTRACT

This article presents observations pertaining to expressive visual design for computer music, focusing in particular on real-time integration of graphics and audio. The author describes specific projects as examples supporting a set of design principles that range from "user-oriented" to "aesthetic" and other observations. Examples include audio visualization, game-like interfaces, and mobile musical instruments.

We operate on multiple simultaneous modes of sensory input, including hearing (sound), sight (graphics) and touch (interaction). These senses mutually reinforce each other and are essential in deriving expression, meaning and aesthetic appreciation when creating and experiencing media, art and design [1,2]. Of our senses, sight and hearing are most readily describable (and perhaps therefore most easily programmable on a computer). This article focuses on the intersection of graphics and audio and discusses strategies for expressive visual design for computer music.

In the process of designing graphics-intensive computer music software, tools and instruments over the last 15 years, I have collected a set of principles for design and have developed a general philosophy of design. These principles are not intended to be universal (or necessarily original); I have derived them from a sustained, iterative process of designing graphical computer music systems. My observations are targeted toward designers of computer music instruments, apps and audiovisual software and serve to provide some rules of thumb, as well as food for thought. Using examples of software, instruments and audiovisual works, this article aims to illustrate these notions of design. The principles are listed below, categorized into user-oriented, aesthetic and other. I refer to these principles throughout the article.

Ge Wang (professor, computer music designer), Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, 660 Lomita Drive, Stanford, CA 94305, U.S.A. Email: <ge@ccrma.stanford.edu>. Web: <www.gewang.com>.

See <www.mitpressjournals.org/toc/lmj/-/26> for supplemental files associated with this issue.

DESIGN PRINCIPLES

User-Oriented Principles

1. Make it real-time whenever possible.
2. Design sound and graphics in tandem: Neither should be an afterthought; seek salient mappings.
3. Invite the eye—of experts and newcomers alike.
4. Hide the technology: Induce viewers to experience substance, not technology.
5. Do not be afraid to introduce arbitrary constraints.
6. Reinforce physical interaction using graphics (especially on touch screens).

Aesthetic Principles

7. Simplify: Identify core elements; trim the rest.
8. Animate, create smoothness, imply motion: It is not just a matter of how things look but how they move.
9. Be whimsical and organic: glow, flow, pulsate, breathe; imbue visual elements with personality.
10. Have an aesthetic; never be satisfied with "functional."

Other Principles

11. Iterate (there is no substitute for relentlessness).
12. Visualize an algorithm to understand it more deeply and discover new directions.
13. Glean inspiration from video games and movies.

VISUALIZING AUDIO PROCESSES

sndpeek and rt_lpc (2003–2004)

sndpeek began as a personal hacking project to make a simple teaching tool that visualizes waveforms and Short-Time Fourier Transforms (STFTs) in real time, using the microphone input (Fig. 1). Fortuitously, while hacking at an extended family gathering, I discovered that the real-time visual response to sound encouraged small children to "experiment" by vocalizing different sounds, eventually escalating into full-on screaming. Without any prompting from me, they intuited

 doi:10.1162/LMJ_a_00960

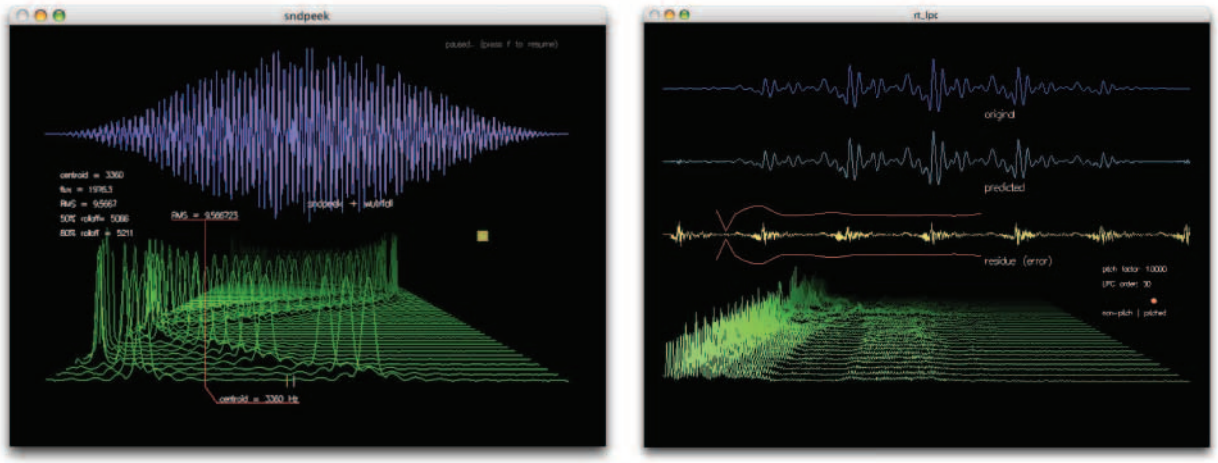

Fig. 1. Real-time audio visualizations: (left) sndpeek's waveform and waterfall plots; (right) rt_lpc visualizing various stages of linear-predictive coding (LPC) analysis and resynthesis. Creating these visualizations led us to more complete understanding of the underlying algorithms (#12: visualize an algorithm to understand it more deeply). (© Ge Wang)

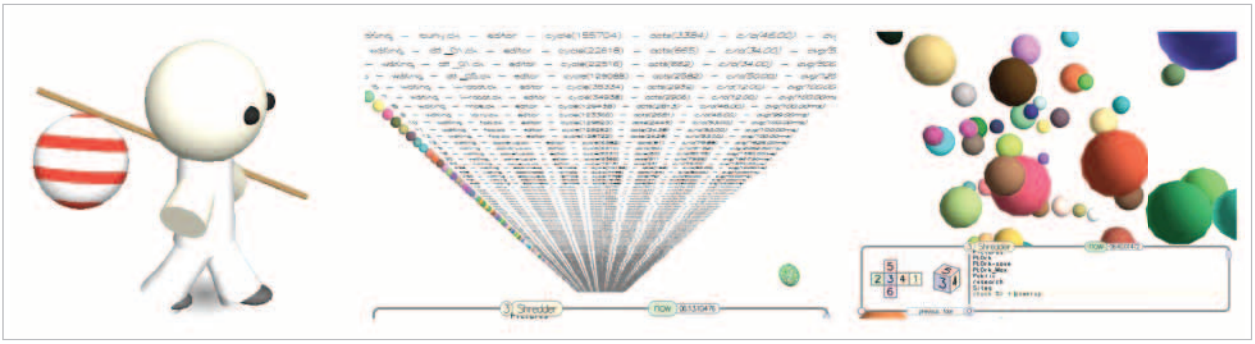

Fig. 2. The Audicle visualizes various elements of the inner workings of a ChucK program; the center and right pane show active and recent processes. (© Ge Wang)

that higher frequencies appeared to the right on the STFT display (#1: make it real-time, and #3: invite the eye). There was even briefly a competition for who could scream the highest (in both loudness and pitch)—before the adults put an end to the enterprise.

In sndpeek, the mapping between sound and graphics is direct and immediate—the time-domain waveforms are simply drawn from each buffer of audio as it arrives from the microphone. A Fast Fourier Transform (FFT) is taken for each buffer, and the magnitude of each FFT bin makes up the spectral display. A sliding history of the STFTs is kept and animated in a scrolling waterfall plot [3]. The tool relies on the smooth animation and motion to convey information (#8: animate, create smoothness, imply motion).

The Audicle (2004–2006)

The Audicle [4] was an ambitious attempt at deep integration between real-time visualization and the inner workings of the ChucK audio programming language [5]. Implemented entirely in C++/OpenGL (as are most of the examples in this article), the Audicle provided multiple views of core elements of the ChucK virtual machine (VM), including timing, processes ("shreds") and scheduling ("shreduling") (Fig. 2). Stats were tracked deep within the ChucK VM and were conveyed to the Audicle for visualization. Originally envisioned to facilitate live coding performances as a type of "program monitor as performance art," the Audicle also contained an audio visualizer, directly drawing from the real-time audio synthesis in ChucK's audio engine, and an animated, physics-based code editor (#1: make it real-time).

Although the Audicle itself was ill fated as an integrated programming environment, it spawned the simpler and much more successful miniAudicle, and it served as the foundation for later laptop orchestra graphical interfaces. It was also a proof of concept for deep integration between graphics and complex audio environments in a real-time context—and the challenges therein.

Converge (2010–2012)

Created as part of an audiovisual composition for the Stanford Laptop Orchestra [6], the Converge visualizer was designed to be a "visual blender" of hundreds of images tagged with location data, timestamps and user descriptions—all collected from people using mobile phones in their daily lives

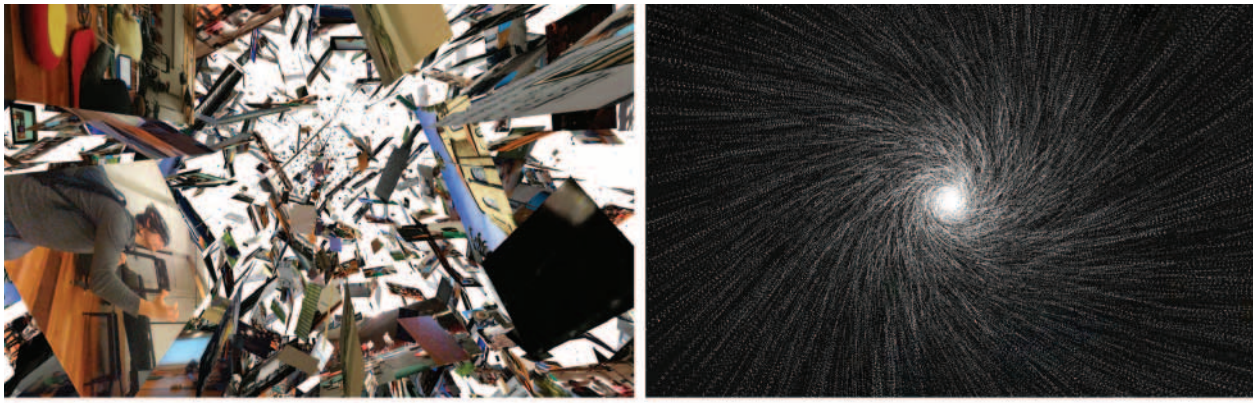

Fig. 3. Converge visualizing and "blending" hundreds of user-generated photos (left); "galactic vortex" visualization, composed of thousands of image fragments (right). (© Ge Wang)

[7] (Fig. 3). The composition was an exploration of the moments/sounds of daily life, memory and the passage of time. Each picture was visualized with a live "count-up" timer that highlighted our perpetual movement away from these past moments (e.g. "8 months, 2 days, 5 hours, 28 minutes, 3 seconds ago . . . 4 seconds ago . . .").

I included an organic visual gesture of exploding each image into 225 image fragments, all subject to a gravitational field that accelerated into a spiraling galaxy-like vortex (#9: be whimsical and organic). The shards could reform into their original images or they could crumple into imploded balls of image fragments (alluding to the imperfections and idiosyncrasies of memory). While these images were gathered prior to the performance and so were not necessarily personal to the audience, they provoked strong emotional responses, possibly because the images were "mundane" moments of everyday life to which anyone could relate (#4: hide the technology).

GAME-LIKE INTERFACES

Non-Specific Gamelan Taiko Fusion (2005)

Non-Specific Gamelan Taiko Fusion, by Perry Cook and myself, was one of the very first pieces created for laptop orchestra [8] and featured a local-area networked and synchronized step sequencer. The ensemble is divided into four sections, each of which can place any of eight timbres, represented by colored squares, on the 8 × 4 sequencer grid (Fig. 4). An animated cursor washes over the sequencer as a human conductor issues instructions to each of the sections regarding timbre and density. The interface is simple (#7: simplify). The animation, while minimal, was designed to make an intrinsically discrete step sequencer feel more fluid.

ChucK ChucK Rocket (2006)

ChucK ChucK Rocket, by Scott Smallwood and myself, is a collaborative instrument inspired by the game *Chu Chu Rocket* [9] (#13: glean inspiration from video games and movies). Players instantiate mouse-like critters onto a game board, directing them with arrows (Fig. 5). As a critter runs over objects, sounds occur. Visually, the critters dart smoothly over the game board; however, the underlying system is a discretely timed grid. Furthermore, all computers in the ensemble are network-synchronized, making complex interlocking rhythms possible. Graphically, an interpolator takes the synchronization signals and computes a velocity

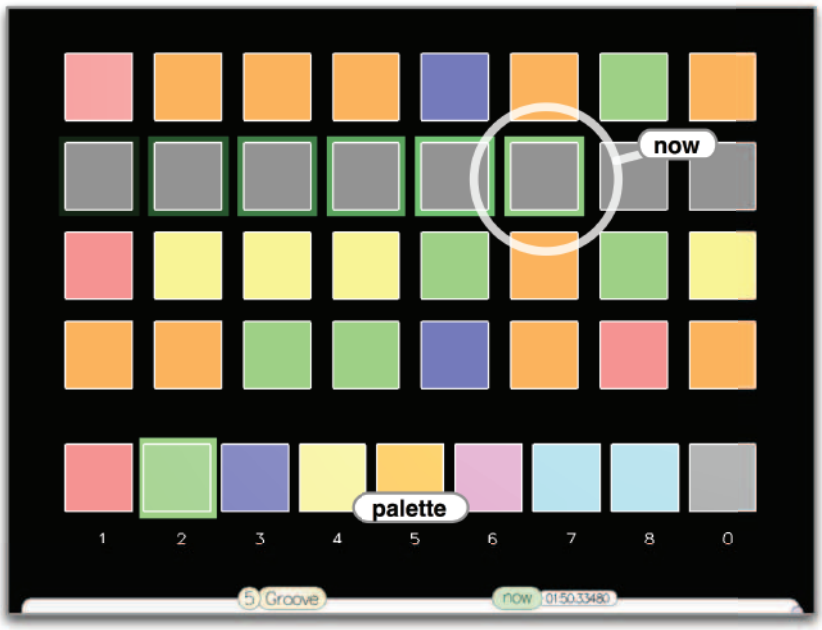

Fig. 4. Interface for *Non-Specific Gamelan.* (© Ge Wang and Perry Cook)

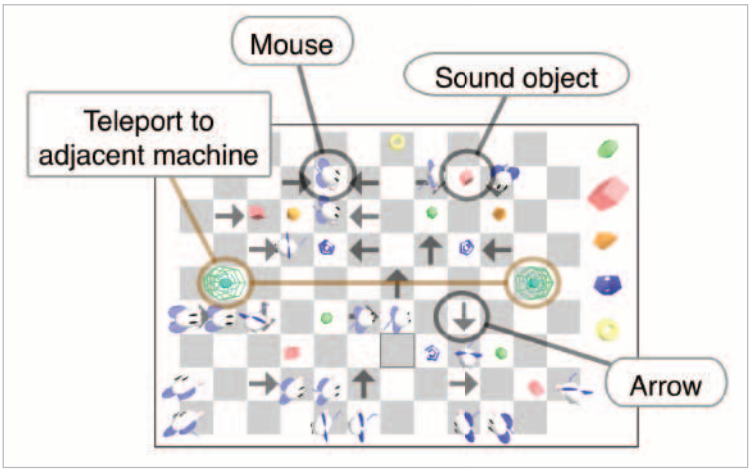

Fig. 5. *ChucK ChucK Rocket* in action: top-down view of game board. (© Ge Wang and Scott Smallwood)

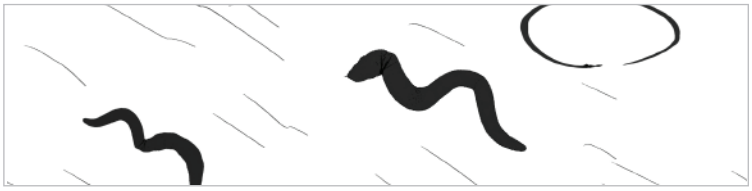

Fig. 6. Levin's *Yellowtail* animates brushstrokes using information from the drawing gesture itself. (© Golan Levin, created using *Yellowtail*)

to animate the movement of each critter, giving the appearance of smoothness (#8: animate, create smoothness, imply motion).

INTERLUDE: INSPIRATIONS

When one is designing visuals for computer music systems, outside inspiration can suggest new functional, aesthetic and technical directions. I believe that inspiration can and should be taken from wherever one may happen to find it. More obvious sources include synthesis or physical modeling parameters, aspects of the sound itself (e.g. waveform, spectral data, features). Less obvious sources of inspiration are video games, cartoons, movies or examples from nature such as how a branch sways in a breeze.

I have sometimes drawn inspiration for design from movies. Consider, for example, "The Sorcerer's Apprentice" in Disney's *Fantasia* [10], where Mickey Mouse (the Apprentice) wields his absent master's magic wand in an attempt to "automate" his chores, leading only to chaos and mayhem. The visuals and animation design for this segment are meticulously and artfully tied to the musical score (by Paul Dukas, 1896), a masterful example of conveying whimsy and magic through visuals and music (as brooms and other household objects take on personalities). At the same time, the Apprentice's role seems symbolic of our own roles as researchers and practitioners of a still nascent technology—it holds our fascination and at times can feel like magic (and sometimes ends up in a mess).

Other sources of inspiration have come from Edward Tufte's insights on information presentation [11], Toshio Iwai's musical games [12] and audiovisual suites like Golan Levin's Painterly Interfaces [13]; Levin's work provides instructive examples of the design of interactive sound and graphics as a single entity (#2: design sound and graphics in tandem) and the foregrounding of substance over machinery. In *Yellowtail* (Fig. 6), Levin imposes the mechanic (#5: introduce arbitrary constraints) of recording the gesture associated with drawing strokes and extrapolating this information to organically animate each stroke while maintaining the essence of how the initial stroke was drawn (#8: animate, create smoothness, imply motion and #9: be whimsical and organic).

VISUAL DESIGN FOR MOBILE MUSIC

Mobile and other touch-screen–based instruments offer yet another dimension for real-time graphics, since the display is also the surface of interaction, presenting unique opportunities to couple visual design with physical interaction design. The visual designs of several musical apps by mobile music startup Smule are discussed below.

Ocarina (2008) and *Ocarina 2* (2012)

Ocarina, designed in 2008 [14], was an exercise to create an expressive musical instrument specifically tailored to the iPhone (Fig. 7). The physical interaction in *Ocarina* includes blowing into the microphone to articulate a sound, using multi-touch to control pitch (via four onscreen virtual holes), and tilting the device to add and control vibrato. *Ocarina* embodies many design principles discussed in this article: It was designed to be visually inviting as an instrument/expressive musical toy (#3: invite the eye). The visual design was also an exercise in reduction (#7: simplify), choosing to show only the functional fingerholes (and not the body of the instrument—a design gesture proclaiming that the phone *is* the instrument) and visualization of breath and spinning particles in *Ocarina 2* that respond to breath input (#9: be whimsical and organic).

Much attention was devoted to the graphical interaction of the onscreen virtual fingerholes—they smoothly expand when a touch is detected. The goal was to make the experience feel responsive and also to compensate (to an extent) for the lack of tactility of the flat touchscreen. Animated fingerholes help inform users, often in their peripheral vision, that they have covered or activated a particular hole (#6: reinforce physical interaction using graphics).

The social aspect of *Ocarina* is presented through a visualization of the earth displaying locations of recent *Ocarina* players around the world and highlighting performance snippets. As accompaniment to the melody, a dual-helix animation emanates from the snippet's origin on the globe

Fig. 7. Animated fingerholes in *Ocarina* reinforce physical interaction by responding smoothly to touch, aimed to compensate for lack of tactile feedback (left); *Ocarina*'s globe: listening to and visualizing other users around the world (right). (© Ge Wang and Smule)

and peacefully swirls into space, evoking a sense of both loneliness and connection. Functionally speaking, this visualization is perhaps completely unnecessary, but aesthetically it seems essential in order to convey a sense of magic (#10: have an aesthetic) and hide the technology (#4). The inspiration for the glowing helix actually came from a visual effect in video games that is often used when a magic spell is cast or when a character gains a new ability (#13: glean inspiration from video games and movies).

Magic Fiddle (2011)

Magic Fiddle was designed specifically for the iPad (Fig. 8) and requires the user to hold the device near the chin and shoulder, like a violin [15]. The bowing interaction was replaced by an interaction that looks like a swirling vortex of smoke when touched, implying an active constant motion (#5: introduce arbitrary constraints and #8: animate, create smoothness, imply motion). The graphics, as in *Ocarina*, were designed to enhance a physical interaction (#6: reinforce physical interaction using graphics) via responsive animations on the strings and in the bowing region. We focused on only the core interactive elements (#7: simplify) and aesthetic elements, including an additively blended neon glow on the strings (#9: be whimsical and organic) and a flowing mist-like effect in the background, which also gave the visual effect of depth while emphasizing the virtual fiddle strings.

The visual and sound design for *Magic Fiddle* proceeded in tandem through many iterations (#2: design sound and graphics in tandem and #11: iterate), where the visual design stemmed from the parameters of the bowed string physical model (based on commuted synthesis), and the graphics guided the features of the sound synthesis, suggesting how glissandi and pizzicati might work. Ultimately, a glowing, neon-like aesthetic was adopted (#10: have an aesthetic).

Magic Piano (2010)

The core interaction in *Magic Piano* is exceedingly minimalistic (perhaps taking principle #7: simplify to an extreme), only involving falling flickering light particles (representing notes) and animated expanding rings in response to touch gestures [16] (Fig. 9). The lack of visible piano keys in this mode was in consideration of both the small touchscreen size and the lack of tactility in distinguishing adjacent keys. The visual design, therefore, tried removing keys altogether and encoded scores in the animated falling particles to be played expressively in time (#5: introduce arbitrary constraints).

This "songbook" mode is by far the most popular mode in *Magic Piano*—itself one of Smule's most popular apps, with more than 80 million users to date. By contrast, the "solo instrument" modes in *Magic Piano* do feature visible piano keys, albeit contorted into various shapes, including a spiral that slowly "breathes," and a linear form that oscillates (#9: be whimsical and organic). These whimsical modes were

Fig. 8. *Magic Fiddle* on an iPad. (© Ge Wang and Smule)

initially design experiments and were left in as solo instruments. (Admittedly, the contorted pianos are some of the most unnecessarily difficult-to-play instruments I have ever designed.)

CONCLUDING REMARKS

Visual design for computer music is as much art as it is science. Said another way, *design* is the art of *articulating* something with technology while conveying something *more* than the technology. The principles here recur (some are much older than any works described here) and have benefited my work and the work of my colleagues, yet they are not meant as guidelines; creativity and invention are always essential. Perhaps these observations can serve as points of reference as we collectively continue to explore the intersection of the sonic and the visual.

Fig. 9. *Magic Piano*'s "songbook" mode (top): user controls timing to play pitches encoded in falling particles; *Magic Piano*'s spiral keyboard mode (bottom): whimsical (and notoriously difficult to play). (© Ge Wang and Smule)

Acknowledgments

Thanks to Perry R. Cook for his mentoring on interaction and aesthetics; to Philip Davidson, Spencer Salazar, Mattias Ljungström, and collaborators at CCRMA, Princeton and Smule; to Jonathan Berger and Chris Chafe for encouraging me to write this article; and to Madeline Huberth for helpful suggestions in crafting this article.

References

1 M. Chion, *Audio-Vision: Sound on Screen* (Columbia Univ. Press, 1994).

2 L. Marks, *The Unity of the Senses: Interrelations among the Senses* (Academic Press, 1978).

3 A. Misra, G. Wang and P.R. Cook, "SndTools: Real-time Audio DSP and 3D Visualization," in *Proceedings of the International Computer Music Conference* (2005).

4 G. Wang and P.R. Cook, "Audicle: A Context-Sensitive, On-the-Fly Audio Programming Environ/mentality," in *Proceedings of the International Computer Music Conference* (2004) pp. 256–263.

5 G. Wang, "The ChucK Audio Programming Language" (PhD dissertation, Princeton Univ., 2008).

6 G. Wang, N.J. Bryan, J. Oh and R. Hamilton, "Stanford Laptop Orchestra (SLOrk)," in *Proceedings of the International Computer Music Conference* (2009) pp. 505–508.

7 J. Oh and G. Wang, "Converge: An Omni-Biographical Composition," *Computer Music Journal Emile* Vol. 9 (2011).

8 S. Smallwood et al., "Composing for Laptop Orchestra," *Computer Music Journal* 32, No. 1, 9–25 (2008).

9 Chu Chu Rocket (Sega Dreamcast) (Sega, 1999).

10 "The Sorcerer's Apprentice," in *Fantasia*, James Algar, dir. (Walt Disney, 1940).

11 E. Tufte, *The Visual Display of Quantitative Information* (Cheshire, CT: Graphics Press, 2001).

12 T. Iwai, "Images, Music, and Interactivity—the Trace of Media Art" (keynote address), in *Proceedings of the International Conference on New Interfaces for Musical Expression* (2004).

13 G. Levin, "Painterly Interfaces for Audiovisual Performance" (MS thesis, MIT Media Laboratory, 2000).

14 G. Wang, "Ocarina: Designing the iPhone's Magic Flute," *Computer Music Journal* 38, No. 2, 8–21 (2014).

15 G. Wang, J. Oh and T. Lieber, "Designing for the iPad: Magic Fiddle," in *Proceedings of the International Conference on New Interfaces for Musical Expression* (2011) pp. 197–202.

16 G. Wang, "Game Design for Expressive Mobile Music," in *Proceedings of the International Conference on New Interfaces for Musical Expression* (2016).

Manuscript received 1 January 2016.

GE WANG *is assistant professor at Stanford University's Center for Computer Research in Music and Acoustics (CCRMA). He specializes in computer music design—researching programming language and software design for music, interaction design, mobile music, laptop orchestras, aesthetics of music technology design, and education at the intersection of engineering, art and design. Wang is the author of the ChucK music programming language, the founding director of the Stanford Laptop Orchestra (SLOrk), the cofounder of Smule (reaching over 125 million users), the designer of the iPhone's* Ocarina *and* Magic Piano, *and a 2016 Guggenheim Fellow.*