Ge Wang
35 Olden St.
Princeton, NJ 08544
(609) 273-6595
gewang@cs.princeton.edu

November 25, 2006

CCRMA Search Committee
Department of Music
541 Lasuen Mall
Stanford University
Stanford, CA 94305-3076

Dear Search Committee,

With great excitement, I submit this letter and material in application for the junior faculty position at CCRMA. Enclosed is my CV, both in the "traditional" format as well as in a re-factored, more detailed "portfolio" version to convey the primary aspects of my research, teaching experiences, and musical works. I hope you find the material interesting to peruse and useful in your evaluation of my candidacy.

I'd like to propose an informal vision of what I would strive to contribute to CCRMA as a new faculty: *In the service of computer music research and pedagogy: computer science theory and practices, cutting-edge software system design, and new performance paradigms.* This "vision" includes and goes beyond sound synthesis and software engineering; it embraces a broad spectrum of topics such as new computer music system design, programming language design and compiler construction, user interface design (which ultimately informs nearly everything I do, including programming language design), human-computer interaction, computer graphics and visualization (of anything potentially useful), audio analysis and music information retrieval, software design and implementation techniques (design patterns and "best practices"), signal processing theory and application, and finding new integrated pedagogical approaches where technology, music, and performance are taught and practiced as a single, naturally intertwined entity. It is my firm belief that a continued and fundamental exploration of these areas has the unique potential to lead the computer music field towards new and profound directions in research and pedagogy.

I refer you to my research and teaching letters, vitae, and portfolio for details. I sincerely hope to bring all my expertise and experience to CCRMA, to collaborate with the wonderful and unparalleled faculty and graduate students there, and to continue developing new ways to benefit research and pedagogy in computer music. Thank you!

Sincerely,

Research

(Please also see portfolio under "research")

As I have attempted to show in my CV and portfolio, I work with a range of topics that fiercely interest me and in which I have first-hand expertise, ranging from audio programming languages and systems (*ChucK* compiler and virtual machine, the *Audicle*), software user interface design (*Audicle*, *TAPESTREA*), audio analysis and transformation (*TAPESTREA*), to new music performance ensembles and techniques (*PLOrk*, *live coding, audio over networks*), visualization (*Audicle*, *sndtools*), and new methodologies for teaching computer music (*PLOrk classroom*, *ChucK*, *sndtools*). I deeply enjoy conducting research in computer music (anything that's fun, interesting, or potentially important) and I thoroughly love teaching computer music, computer science, and computer-mediated composition & performance. I actively compose and perform – including for multi-channel tape, live coding performances, and in new ensembles such as PLOrk: Princeton Laptop Orchestra.

My PhD dissertation describes the research, theory, engineering, and applications of the *ChucK* audio programming language, and *the Audicle*, a graphical environment for on-the-fly programming and, more generally, for visualizing the audio programming process. In my thesis, I argue for the benefits of an audio programming language that gives the programmer highly precise control over time and concurrency (we call this *strongly-timed*), a straightforward syntax, and the ability to program "on-the-fly". The language unifies control over low-level DSP timing and higher-level "musical" timing into a single programming framework and provides a dead-simple (and powerful) concurrent programming model. ChucK is an ongoing experiment in designing and building a computer music programming language completely "from the ground-up", in which flexibility and readability of code is favored over raw performance. Indeed, while the implementation is currently un-optimized and resource-intensive, the language syntax and semantics are flexible for veteran programmers to use, and yet easy enough for programming novices to learn quickly. The ChucK community is highly enthusiastic and rapidly growing, and more institutions are adopting ChucK into their classrooms. At Princeton, we have extensively used ChucK as a teaching tool and as a platform for building new pieces in the Princeton Laptop Orchestra ever since the ensemble's inception in 2005; more than twenty ChucK-enabled PLOrk pieces have premiered to date. ChucK has twice won the ICMA Best Presentation Award (2003 and 2004), and has won the 2004 ACM Multimedia Open Source Competition. Presently, we have just begun to explore a new dimension: combining real-time synthesis and audio analysis/music information retrieval support into the same programming framework. This is still future work – but I believe it can provide powerful ideas for the way we write programs in the future for sound synthesis and analysis.

My intense interest in visualizing anything and everything that might be potentially useful (or pretty to look at) drove us to design and implement the Audicle (started in 2004, see portfolio for screenshots) to visualize the inner workings of ChucK as it runs a user program. It includes a "smart" editor, visualization and control for ChucK processes, inter-process timing, real-time audio, and more. The goal of the Audicle is not

only to visualize sound, but to also provide *insight* about audio programs as we write them. The Audicle also serves as a platform for building completely customizable user interfaces – it is the foundation for several full-fledged PLOrk instrument interfaces and a workbench for prototyping new user interfaces.

Also in 2004, I helped to originate the *TAPESTREA* system, with Ananya Misra (project leader) and Perry Cook. TAPESTREA is a unified framework for interactively analyzing, transforming and synthesizing complex sounds. It provides the means to identify points of interests in a sound, and extract them into reusable templates, which can then be *re-composed*. My contributions include leading the design and development of the user interface, architecting the synthesis engine, co-designing the core system, and integrating ChucK into TAPESTREA as a full-fledged scripting language, as well as being involved in many other aspects of the project. Our work on TAPESTREA recently received the 2006 ICMA Distinguished Paper Award (for our paper "Musical Tapestries: Re-composing Natural Sounds"). With Perry and Ananya, I also composed *Loom*, an 8-channel tape piece using TAPESTREA. This piece was premiered at ICMC 2006.

In 2005, I became one of the founding developers and instructors of *Princeton Laptop Orchestra* (PLOrk). I helped to create the initial core curriculum, which teaches ChucK, Max/MSP, live coding, compositional issues, performance techniques, and new paradigms for conducting and synchronization. Perry and I wrote one of the first pieces for PLOrk, titled *Non-specific Gamelan Taiko Fusion*. I later developed a series of collaborative graphical user interfaces/instruments using ChucK and the Audicle, including several game-based pieces with Scott Smallwood, as well as four additional compositions. Currently (fall 2006), I co-direct the ensemble with Perry as part of a unique course that tightly integrates composition, performance, and computer science.

As mentioned above, I am incredibly interested in finding new ways to present information about pretty much anything related to computer music. In 2004-2005, I, with Perry, Ananya, and others, implemented *sndpeek*, a real-time spectrum/waveform visualization program. In addition to achieving memorizing popularity with kids of almost any age (they love screaming at and “looking” at the sound of their voice), sndpeek has proven to be a fun tool for showing sound and basic spectral features. In 2005, John Chowning worked with us to develop an enhanced version of sndpeek in a new premier of *Stria*. sndpeek was followed by *rt_lpc*, a real-time LPC analysis/resynthesis program with visualization, and *rt_pvc*, a real-time phase vocoder.

Another area of interest to me is interactive audio and musical performance over networks. I was responsible for implementing the networking system that supported *Gigapop Ritual* (2003), an interactive performance between Princeton and McGill University at NIME 2003. The system transmitted two-way uncompressed audio, MIDI data, and one-way uncompressed video over Internet2 and CA2Net. More recently, I’ve explored new techniques for synchronizing computer music ensembles (e.g. PLOrk) over local-area. Overall, I am an enthusiastic advocate of close collaboration in working on large, cool research projects, and I am fully confident and excited about exploring (and forging if helpful) new areas in computer music research.

Teaching

(Please also see portfolio, under "teaching")

My teaching philosophy includes the following ideas: teaching-by-example, hands-on, encouraging openness and creativity, knowledge founded on an inspired understanding of the underlying technology, and technology informed by aesthetics and "desires" of music and hearing. I am also extremely interested in "naturally" integrated classrooms that combine technology, computer music synthesis, and performance-related issues.

I would like to describe two first-hand experiences that seem relevant. The first is a graduate course I taught at Dartmouth College in the Electro-Acoustic Music program, which I called "In the Service of Electro-Acoustic Music: Digital Signal Processing + Software Design/Implementation Techniques". The seminar was aimed at technology-minded composers; it was to provide an introduction to DSP for computer music, as well as a hands-on, "in-the-trenches" introduction to designing software computer music systems. I designed a curriculum that introduced each DSP concept, such as convolution, filter analysis, Fourier analysis, both from mathematical foundations as well as using practical implementations. We also discussed topics such as object-oriented design, optimization techniques (and when to optimize), design principles and patterns, and building real-time systems. We devoted a full lab day each week to dissecting code, software design, and to forge ahead on implementing a lightweight real-time synthesis system. Overall, I believe the course gave the students knowledge in two crucial areas: why things happen (DSP theory), and how to make it happen (software techniques and "best practices").

A second experience I want to describe is that of developing, teaching, and composing for the Princeton Laptop Orchestra. I was one of the four founding developers of PLOrk (faculty: Dan Trueman, Perry Cook, graduate students: Scott Smallwood and myself). We started in Fall 2005 with no clear idea of how to run, teach, or even survive a laptop ensemble consisting of 15 players, 15 laptops, software, hardware, sensors, and 90 independently addressable loudspeakers. Slowly, painstakingly, perhaps amazingly (and with massive help and enthusiasm from students, faculty, and composers), we are figuring things out. The initial ensemble laid the groundwork for the operation of the PLOrk classroom and established the ChucK programming language as a unique and successful pedagogical tool. The second course focused on playing as an ensemble. A third, currently in-progress seminar/ensemble (which Perry and I are teaching/directing) combines composing for laptop orchestra and performance.

Using ChucK, we have been able to teach audio programming to first-time programmers, who in a matter of weeks were able to create live performances using code. Having taught programming before (in computer science classroom, companies, private tutoring, etc), I noticed a welcoming difference in teaching ChucK in PLOrk: the students made and performed music - and solidly learned programming "on the side", almost as if by accident. Every new concept (e.g. looping, functions, concurrency) we introduced was eagerly received by the students, who immediately saw how they might use it to enhance their music programs or to program things they couldn't before. They were so eager (and

worked so hard) we could hardly keep up feeding them new information. (That inaugural semester we were putting in 50-60 hour weeks to keep things running).

This fall (2006) Perry and I are teaching a new PLOrk graduate seminar titled "Composing for Laptop Orchestra" and things are firing on all cylinders (at least compared to a year ago). We've already held 3 success concerts this semester, created and premiered as many new pieces (15 and counting) as there have existed before, while finding new ways to teach and play with the ensemble. We are thankful for and extremely pleased at the success of this classroom that *truly* integrates technology, computer music, and computer-mediated performances.

Lastly, the following are some courses I could very happily teach.

Graduate seminars:

- *Designing Software Systems (S,M,L,XL) for Computer Music* (designing languages, building compilers, real-time performance systems)
- *Software Engineering Techniques and Strategies for Computer Music* (design patterns, advance OO techniques, optimization, networking, etc.)
- *Topics in Audio Synthesis + Analysis / Music Information Retrieval*
- *Computer Graphics & Visualizing in the service of Computer Music*
- *Designing Software Human-Computer Interfaces* (design and implementation)
- *Composing for New Performance Ensembles and Paradigms*

Undergraduate courses:

- *Computer Music* (programming, composition, performance)
- *Computer Music Synthesis + Analysis: an Introduction*
- *Introduction to DSP + software engineering for Computer Music*
- *Computer Music History*
- *Experimental Integrated Classroom for Computer Music* (like PLOrk)

Thanks you for your time.