Palabras previas

"¡Feliz ChucKing!"

¿Qué significa esta corta frase, dicha por Ge Wang, uno de los autores de este excelente libro? Mi interpretación es que significa explorar y componer sonidos de manera divertida con ChucK, el lenguaje de programación que es la base de este libro. Pero, "ChucKing" es más que eso - es una manera de abordar el aprender a programar con un enfoque en las artes; es al mismo tiempo animado y profundo.

Como muchos otros, a mí me importan las artes visuales y la música por sobre todas las cosas. También estoy interesado en los computadores, pero primordialmente como herramientas para hacer imágenes y ruido. Sin embargo, como todas las veces que fallé en aprender cómo programar computadores cuando tenía entre 10 y 25 años, estuve forzado a aprender código a través de texto - imprimiendo "Hello World" en una pantalla o escribiendo código para calcular números. Una aclaración, me gusta escribir y creo que las matemáticas tienen un valor incalculable para lo que amo hacer, pero las palabras y los números nunca son el foco de atención. Siempre son un medio para llegar a un fin.

¿Qué pasaría si pudiera aprender a programar ayudado por lo que más me importa? ¿Aprender a programar haciendo imágenes y ruidos? Antes de que los computadores se convirtieran en las extraordinarias máquinas multimedia de la actualidad, la mayor parte de las personas usaban computadores para trabajar exclusivamente con texto. Los estudiantes que estaban más interesados en imagen y sonido no eran capaces de aprender a programar a través de la persecución de sus pasiones. Afortunadamente,

esto ha cambiado y ahora los computadores (desde los teléfonos móbiles hasta los supercomputadores) pueden generar imágenes, sintetizar sonido y mucho más. Desafortunadamente, la mayor parte de las clases para aprender a programar siguen siendo de la misma forma que hace 40 años - aprender a programar fuerza a todos, artistas visuales y músicos por igual, a adaptarse a las rígidas restricciones de usar caracteres alfanuméricos como entrada y salida.

Me esforcé en aprender código de la manera tradicional. Durante la última década, he enseñado a programar a través de una nueva plataforma de programación que coinventé con Ben Fry. En MIT en el año 2001, empezamos a desarrollar un entorno y lenguaje de programación llamado Processing. Processing fue creado para que aprender a programar por primera vez y alienta a "bosquejar" con código. Lo más importante de Processing es que puedes aprender todos los fundamentos de programar, pero a través del trabajo con medios visuales dinámicos - por ejemplo, con dibujo, color y animación. En el momento en que empezamos a trabajar en Processing, no sabíamos que Ge Wang, en ese entonces un estudiante de posgrado en Princeton, estaba haciendo lo mismo en el dominio de la música. A través de su lenguaje de programación sobre la marcha, ChucK, se aprende a programar a través de la creación de onsido.

Uno de los primeros programas que verás en este libro va al grano:

```
SinOsc s => dac;
440 => s.freq;
1 :: second => now;
```

Este programa crea un tono puro por un segundo: es el equivalente en software de tocar una tecla de piano por primera vez. Esta es una

extraordinaria manera de aprender a programar; invita a preguntas emocionantes. ¿Qué es este extraño símbolo =>? ¿A qué se refiere el número 440? Las respuestas a estas preguntas abren un nuevo mundo; una nueva manera de pensar en cómo crear sonido y música y de forma simultánea aprender los fundamentos de la programación. Desde este programa, se abre un nuevo mundo de posibilidades.

Me emocionó leer Programación para músicos y artistas digitales. Con una multitud de ejemplos bien explicados en el fascinante lenguaje ChucK, los lectores aprenden de manera activa y comprometida. No puedo imaginarme un grupo de gente más capacitada y hábil para escribir cómo aprender a programar a través de la creación musical. Ajay Kapur, Perry Cook, Spencer Salazar y Ge crearon ChucK y desarrollaron la manera en que es enseñado. Después de una década de experiencia con ChucK en la sala de clases y una gran experiencia previa a eso, este libro deja la vara imposiblemente alta. Espero que disfrutes aprender a programar de forma "Chuckian" y "¡Feliz ChucKing!".

CASEY REAS

PROFESOR ASOCIADO

DEPARTAMENTO DE ARTES MEDIALES Y DISEÑO EN UCLA

COCREADOR DEL LENGUAJE DE PROGRAMACIÓN PROCESSING

Prefacio

¡Bienvenido a ChucK!

Tendemos a realizar nuestro mejor esfuerzo cuando seguimos nuestros reales intereses. ChucK fue creado porque amamos de manera genuina tanto la música como la programación - y para cada cualquier persona que quiera hacer música con computadores (o que quiera aprender a hacerlo). Como el creador y el diseñador principal de Chuck, estoy convencido firmemente que programar es (o debería ser) una tarea creativa en sí misma. Debería ser entretenida, expresiva y gratificante. Y crear música a través de la programación, es doblemente asombroso.

Empecé a trabajar en ChucK en el año 2002, habiendo empezado el programa de doctorado en Ciencias de la Computación de Princeton University un año atrás. Si la música rock fue mi drogra de entrada a hacer música (y mi alma mater de pregrado Duke University mi iniciación a la programación), entonces Princeton fue donde empecé a combinar ambos elementos. Aunque no lo podía articular en ese momento, estaba atraído por la elegancia de ciertas características de los lenguajes de programación, y aspiraba a crear cosas, cosas programables con software, que empoderaran a a otros para hacer música, pero de manera estéticamente elegante y entretenida. Quería rockear, y ayudar a otra gente a rockear - con el computador.

Perry R. Cook era mi tutor (una de las mejores cosas que me han pasado a nivel personal y profesional); su trabajo en diseño de interacción física y su Synthesis ToolKit (STK) fueron una gran inspiración. También su personalidad divertida, fantástica e imaginativa ayudó a definir el tono y alentó la libertad en la exploración. Todavía recuerdo cuando le mostré a Perry mis ideas iniciales de un "nuevo lenguaje de programación para música" cuya cohesión se basaba en el operador ChucK (que se ve así =>, lo verás mucho en este libro), y a Perry diciendo, "Bueno, eso parece una locura. ¡Hazlo!"

HECHO: las porciones iniciales de mi escrito permanecieron en el pizarrón de la oficina de Perry por los siguientes cinco años, aunque luego descubrí que esto era debido a un desperfecto del pizarrón que lo hacía casi imposible de borrar.

Poco tiempo después, y de forma casual, diseñé los mecanismos de tiempo y concurrencia (ambos intrínsecos al funcionamiento de ChucK) en el lenguaje naciente. Digo de forma casual porque ambos componentes compenzaron como divagaciones, pero una vez que resolví cómo estos mecanismos podrían funcionar en conjunto, esto resonó tan fuertemente con mi mente que me empeciné en lograr construirlos. Así empezó una travesía para hacer Chuck, que sorprendentemente para mí, se mantiene tan fuerte como nunca más de una década después, y que ha involucrado a algunos de los colaboradores más locos y más maravillosos que he conocido.

Ajay Kapur era un estudiante de pregrado loco y laborioso en el laboratorio de Perry, construyendo la eTabla, la eSitar, el eDholak (un tambor indio para dos personas), y que continuaría con la creación de orquestas completas de máquinas robóticas. Maravillosamente apoyando a Chuck desde el principio, Ajay fue pionero de nuevas formas de enseñar música por computador y diseño de interacción con Chuck. Actualmente enseña en CalArts y es el (aún loco) decano asociado de tecnología musical.

Un día en el año 2004, un estudiante de pregrado de Princeton llamado Spencer Salazar me envió un email, confesando que amaba programar y construir cosas, y preguntó si podía ayudar con el proyecto ChucK. Contestar que sí cambió a ChucK para siempre, porque Spencer resultó ser un gran programador e innovador. Además de añadir soporte para palanca de mando, ratón, teclado, WiiMotes (a través de interfaces HID,

que son tratadas en el capítulo 10), Spencer escribió el miniAudicle, el entorno de desarrollo usado en este libro. ¡Y todo esto fue solo en el primer año! Soporte serial, ChuGins, Chugens y Chubgraphs se encuentran entre sus adiciones más recientes (cubiertas en el apéndice H). Spencer se encuentra actualmente estudiando el programa de doctorado en el CCRMA de Stanford University.

HECHO: Spencer y yo somos ávidos fans de StarCraft, y de forma semi-regular somos humilados en línea por jugadores con habilidades reales. Ajay una vez jugó StarCraft por 8 horas seguidas en un viaje a Montreal, y no ha vuelto a jugar desde entonces (hasta donde llega mi conocimiento). Perry, hasta donde yo sé, nunca ha jugado este juego.

Ahora probablemente ya estás haciéndote una idea del tipo de personas locas (y relajadas) que han trabajado para hacer ChucK posible. La lista no termina ahí. Philip Davidson y Ari Lazier contribuyeron con grandes secciones de ChucK; Phil y yo incluso construimos un prototipo de un ambiente de desarrollo animado en tiempo real llamado el Audicle (un predecesor en bruto al MiniAudicle). Ananya Misra y Matt Hoffman contribuyeron generosamente con grandes porciones de nuevas funcionalidades, mientras que Adam Tindale escribió un manual de ChucK. Al mismo tiempo, sabios individuos como Paul Lansky, Roger Dannenberg, Andrew Appel y Brian Kernighan (todos ellos creadores de lenguajes de programación) dispensaron generosamente su sabiduría en torno a la dirección de ChucK.

La comunidad en línea de ChucK se formó entre los años 2003 y 2004, y empezamos rápidamente a lanzar correcciones de errores y nuevas características (y nuevos errores). Era mágico ver cómo gente que no conocíamos generosamente contribuía a un proyecto de código abierto y

una comunidad como esta. Un individuo particularmente notable es Kassen Oud, quien ha guiado a un sinnúmero de nuevos usuarios, respondido a innumerables preguntas y en general ha distribuido paz y buena voluntad a todos los usuarios de ChucK por más de 10 años. (Kassen tiene el crédito de SporksPerson, moderador del foro y ChucKian Sherpa, en nuestro sitio web). Nunca he conocido a Kassen, pero intuyo en él a alguien de un espíritu gentil.

Existe una regla del desarrollo de software de "The Mythical Man-Month" (un gran tratado en cómo los humanos crean software) que dice: planea desechar uno. La idea detrás de esto es que cuando construyas algo nuevo, la primera implementación a menudo no es la adecuada en términos de arquitectura. Uno no debiera temer a empezar todo de nuevo, armado del conocimiento ganado del primer intento (esto es realmente una muy buena regla para muchas cosas en la vida). En el año 2005, remodelé completamente el núcleo de ChucK para añadir soporte de arreglos y objetos, y también para hacer más fácil una multitud de otras características. Esto resultó ser una buena movida, para la (totalmente loca) Princeton Laptop Orchestra (PLOrk) que empezó ese otoño, conducida por el magnífico Dan Trueman y Perry, con Scott Smallwood y yo como profesores asistentes. ChucK fue usado como la herramienta principal de programación y enseñanza.

Este fue otro momento transformador, porque fue la primera vez que se enseñó ChucK en un formato tradicional de sala de clases, por un semestre completo, a 15 estudiantes universitarios de pregrado de primer año sin conocimientos previos de programación. También resultó ser la primera vez que alguien haya enseñado a una orquesta de computadores, así que literalmente fuimos inventando sobre la marcha. Fue deliciosamente angustiante y estimulante porque no estuvimos guiados ni

limitados por reglas establecidas. El curso resultó ser un éxito rotundo, ChucK estuvo a prueba como nunca antes, y sabíamos que podíamos enseñarle ChucK a casi cualquier persona con el justo interés y pasón por aprender.

Hechos increíbles siguieron ocurriendo, ya que Rebecca Fiebrink llegó en el año 2006 a empezar su doctorado, y yo encontré en ella a una colaboradora muy querida e increíble. Durante el año que coincidimos en Princeton (antes de que yo me fuera a enseñar en Stanford University), Rebecca y yo logramos una cantidad asombrosa de cosas relacionadas a ChucK, incluyendo el Small Musically Expressive Laptop Toolkit (Chuck SMELT), además de la arquitectura de los Unit Analyzers en ChucK. A través de todo esto, puedo decir que las cosas positivas ocurren cuando (y realmente solo porque) hay gente estupenda involucrada, y ChucK definitivamente encarna las personalidades de quienes lo construyen.

En el año 2007 (y mientras todavía estaba trabajando en mi disertación de doctorado sobre ChucK) me uní como profesor al Center for Computer Research in Music and Acoustics (CCRMA, pronunciado como "karma") de Stanford University. El director del CCRMA Chris Chafe (otra alma bondadosa) ya estaba enseñando ChucK en sus cursos, y yo empecé a hacer lo mismo en la reción creada Stanford Laptop Orchestra (SLOrk), completa con sus arreglos de parlantes construidos con cuencos de ensalada de IKEA. (En el presente, con Spencer y yo ambos en el CCRMA, se convirtió en la central para la investigación y desarrollo de ChucK).

Como si las cosas no fueran lo suficientemente alocadas, cofundé un emprendimiento en el 2008 llamado Smule, con el fin de acercar la creación musical mediada por computador a una audiencia general en aparatos móviles, y en sinergia con la investigación académica de música

por computador. Spencer fue un desarrollador fundador y Perry un consejero fundador. ChucK se convirtió en nuestra plataforma de prototipado rápido y en parte del núcleo de audio. Diseñé Ocarina, una aplicación que transformaba el iPhone en un instrumento expresivo tipo flauta, e incluso le permitía a los usuarios escuchar al resto a lo largo del mundo, y usaba a ChucK como el motor de audio e interacción. A la fecha, ChucK está en más de 10 millones de iPhones a través de las aplicaciones Ocarina y Ocarina 2 para iPhone.

Mientras tanto, más y más gente relacionadas a instituciones académicas están programando con Chuck. Por ejemplo, Ajay ha confeccionado un currículum completo en CarLarts para enseñar tecnología musical con ChucK (programación de audio), Processing (programación visual) y Arduino (microcontrolador). A través de una iniciativa con fondos del NSF, Ajay y Perry se han embarcado en la creación de un currículum basado en ChucK orientado a los estudiantes de arte digital. Se han apoyado del grupo de autores para escribir este libro.

POR QUÉ USAMOS CHUCK

Como usuarios devotos de ChucK, cada uno tiene sus propias razones para usarlo. A Perry le encanta ChucK porque necesita programar frecuentemente para mantener la cordura, y ChucK es "el único lenguaje que me deja aparecerme por un par de segundos, o tanto tiempo como quiera, y lograr hacer algo satisfactorio. Si tengo una idea, la puedo poner a prueba de forma inmediata en ChucK."

A Spencer le gusta ChucK porque le permite expresar en código de forma más clara y concisa ideas estéticas complejas y detalladas. "Más que

cualquier otra herramienta, ChucK me da la habilidad de controlar procesos computacionales y los medios para sintetizarlos de forma satisfactoria".

Como artista robótico, Ajay trabaja con MIDI de controladores personalizados y para instrumentos robóticos personalizados. "El poder de ChucK ha sido el multithreading y cómo puedo tener múltiples procesos corriendo con datos de sensores, control de actuadores y procesos de composición, ¡y todos a distintas velocidades! Es por esto que uso el lenguaje para enseñar ciencias de la computación a artistas... mostrándoles un paradigma de lenguaje del futuro..."

Le preguntamos a más gente por qué usan ChucK, y aquí está lo que dijeron:

ChucK sugiere una mentalidad de aproximación a la música, y al diseño de música, la experimentación con música, y el aprendizaje de programación... Me gusta la manera en que ChucK me hace pensar, y yo estaba realmente deprimida con la programación de música antes de conocer ChucK.

Rebecca Fiebrink

Profesora de Gráficas e Interacción

Departamento de Computación

Goldsmiths, University of London

Con ChucK, puedo fácilmente pensar en el flujo del tiempo y en cómo combinar múltiples capas musicales simultáneamente. Es el único lenguaje que conozco que es inherentemente contrapuntual, y es por naturaleza

extraordinariamente musical (y entretenido) para programar.

Dan Trueman

Profesor de Música, Princeton University

Cuando todo funciona de la manera en que se supone que tiene hacerlo, cuando mi arreglo espontáneo de jerga computacional se transforma en una composición musical, es realmente una experiencia asombrosa. La habilidad de controlar duración y altura con bucles, números enteros y notación de frecuencia me envió en un viaje poderoso. (Sobre aprender a programar con ChucK)

Anna Wittstruck

Princeton University, clase del año 2008

Y ahora estamos aquí - y tú estás aquí - con un libro sobre ChucK. Pero realmente este es un libro sobre aprender a lograr que el computador te ayude a expresarte, sónicamente, interactivamente y musicalmente. Para este fin, te dejaré con unos consejos acerca de aprender ChucK.

- No entres en pánico: cuando las cosas no funcionan como lo esperas, no te desalientes. ChucK es un lenguaje para experimentar con sonidos, música e interración, e incluso el programador con más experiencia cometerá errores en su código. No pienses demasiado en el hecho de que estás programando, y deja que tu lógica, curiosidad o visión creativa te guíe.

- Persevera: no importa cuál sea tu experiencia en programación (inlcuso si es nula; incluso si piensas que es aún menor) - mientras tengas interés, puedes aprender ChucK. Además, programar para

hacer sonidos es una de las mejores y más rápidas formas de aprender a programar: para empezar, puedes escuchar de forma inmediata los resultados de tus acciones.

- Haz los ejercicios sugeridos a lo largo del libro y no tengas miedo de experimentar: puedes empezar modificando los números en los programas de ejemplo y escuchando el resultado, y puedes empezar a entender cómo funciona el código, incorporar tu propia lógica e imaginación al código, o incorporarlo a tus propias creaciones.

- Escucha: se trata de hacer sonido, ¡así que usa tu oído a menudo!

Para finalizar, siento una combinación de alegría (porque la gente está usando ChucK para ser creativa, expresiva y hacer música) y de perpetuvo pavor / consternación (yo sé dónde están los errores en la implementación). Esto se suma a una gran gratitud a todos quienes han valientemente programado una línea de ChucK, a quienes han creado un nuevo sonido, interacción, o una nueva pieza musical, y a todos los que quieren aprender.

Así que como decimos en la tierra ChucK (donde ahora te encuentras): el tiempo es ahora. No hay mejor tiempo que el presente. 'Diviértete y haz música fantástica y alegre!

En representación de mis coautores,

GE WANG

PROFESOR ASISTENTE

CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS (CCRMA)

STANFORD University

CREADOR DE CHUCK, COFUNDADOR DE SMULE

Agradecimientos

Los autores le agradecen a los editores de Manning que trabajaron estrechamente con nosotros para mejorar este libro de introducción a todo tipo de gente a ChucK, especialmente a Susanna Kline, Bert Bates y Jeff Bleiel, y a los muchos otros que trabajaron con nosotros en la producción de este libro.

También le agradecemos a los siguientes revisores externos por todos sus comentarios y sugerencias útiles: Edward Borasky, Brent Boylan, Matthew Dickinson, Boyan Dzhorgov, Pierre Jolivet, Hector Lee, James Matlock, Ari Pappas, Patrick Regan, Gabriel Rey-Goodlatte, Alvin Scudder, Sergiy Seletskyy, Nathan Smutz, David Sumberg, Danny Vinson, Dan Warren y Stephen Wolff. Agradecimientos especiales a Doug Sparling, quien hizo una revisión técnica detallada del manuscrito justo antes de que empezara la producción.

Como Ge mencionó en el prefacio, es mucha la gente que ayudó con la creación, desarrollo y maduración de ChucK, pero debemos agradecer particularmente a Ari Lazier, Philip Davidson, Dan Trueman, Scott Smallwood, Rebecca Fiebrink, Ananya Misra, Kassen Oud y Chris Chafe.

También le agradecemos a las generaciones actuales de estudiantes de LOrk (orquesta de computadores), profesores asistentes, y a los compositores que han usado ChucK con nosotros y lo han convertido en

un mejor lenguaje de programación. Estos incluyen a los miembros de la Princeton Laptop Orchestra (PLOrk), Stanford Laptop Orchestra (SLOrk), Stanford CCRMA, y a la comunidad de usuarios de ChucK.

También le agradecemos a los estudiantes de CalArts y a los profesores asistentes que nos han ayudado a corregir errores en nuestro currículum de ChucK, los códigos de ejemplo, este libro y en nuestro curso masivo en línea de Chuck. Estos incluyen a Jordan Hochenbaum, Owen Vallis, Dimitri Diakopoulos, Ness Morris, Bruce Lott y Rodrigo Sena.

Agradecimientos especiales a Casey Reas de UCLA, cocreador del lenguaje de programación Processing, por contribuir con las palabras previas de nuestro libro.

Este libro fue desarrollado en parte con apoyo de la National Science Foundation bajo la Grant No. 1140336. Cualquier opinión, hallazgo y conclusiiones o recomendaciones expresadas en este material son de los autores y no necesariamente reflejan la visión de la National Science Foundation.

Finalmente, le agradecemos a todos los usuarios de ChucK a través de los años, a lo largo del mundo, por compartir en los foros de desarrolladores y usuarios de ChucK, por acompañar a nuestra banda de programadores artistas, o artistas programadores, sugiriendo, reclamando y alabando el lenguaje, y por ayudarnos a a hacerlo aún mejor en el proceso.

Acerca de este libro

Escribimos este libro para enseñar programación en ChucK a

programadores sin experiencia. Por esto, empezamos desde el principio, así que no tengas miedo. Enseñaremos a programar a través de ejemplos musicales. Creemos que ser capaz de "escuchar" lo que estás programando te ayudará a entender los conceptos claves, además de hacer el proceso más ameno. Si eres un programador con experiencia, puedes saltar algunos de los primeros capítulos según tu nivel, pero ten en cuenta que Chuck es bastante diferente de otros lenguajes. Te prometemos que hay enseñanzas para todos aquí.

Si eres un programador avanzado que ya está familiarizado con otro lenguaje, "¿Qué es lo diferente de ChucK?", más adelante en esta materia, te da una estrategia para aproximarte a ChucK desde tu nivel.

Cómo está organizado el libro

A través de los capítulos hay ejercicios de composición sugeridos para expandir lo que has aprendido. De una forma, puedes ver cada capítulo como "desbloquear" nuevas partes del lenguaje que puedes usar en tus composiciones.

En el capítulo 0 (los científicos de la computación casi siempre enumeran desde cero), te diremos cuál es la diferencia entre ChucK y otros lenguajes, y cómo nosotros, y muchos otros lo han usado en múltiples proyectos musicales y artísticos.

Los siguientes dos capítulos son para el programador novato, introduciendo conceptos clave necesarios para ser capaces de empezar a programar. El capítulo 1 empieza con los fundamentos de ciencias de la computación, lenguajes y ChucK, incluyendo variables, tipos, expresiones condicionales y estructuras de bucle. Por supuesto, también hacemos

música y sonido usando ChucK. El capítulo 2 introduce bibliotecas (herramientas) internas a ChucK, y muestra cómo números aleatorios y expresiones matemáticas pueden ser usados para realizar programas y canciones con mayor expresividad.

En el capítulo 3 introducimos los arreglos, y mostramos cómo hacer más fácil la creación, almacenamiento y reproducción de melodías y control de otros parámetros en ChucK.

En el captíulo 4 introducimos cómo usar archivos de sonido en ChucK y cómo puedes usarlos para crear paisajes sonoros e incluso una pieza de música electrónica bailable.

En el capítulo 5 introducimos el concepto de funciones y cómo pueden ser usadas para modularizar y organizar tu código, lo que puede resultar en composiciones aún más expresivas y programas más ordenados.

El capítulo 6 se adentra más profundamente en Unit Generators (UGens), objetos de procesamiento y síntesis de audio incorporados en ChucK. Aquí aprenderás más sobre osciladores, generadores de envolvente, síntesis FM, síntesis de modelamiento físico y efectos de audio. El capítulo 7 continúa esto, introduciendo muchos UGens de síntesis de partículas físicas, modales (resonantes) y UGens de síntesis de partículas.

El capítulo 8 introduce multithreading y concurrencia y aprenderás cómo hacer que tus programas puedan manejar múltiples cosas al mismo tiempo, todo en perfecta sincronía.

El capítulo 9 introduce programación orientada a objetos (POO) y enseña cómo tú puedes crear tus propios Objetos y Clases para usar en tu código.

El capítulo 10 cubre Eventos, que le permiten a los programas de Chuck mandar señales entre ellos. Los Eventos también le permiten a ChucK responder a señales y datos desde el "mundo exterior". Esto nos permite empezar a concebir a ChucK como una herramienta de performance en vivo, introduciendo el uso de tu teclado y ratón para controlar en tiempo real tus composiciones.

El capítulo 11 profundiza en las maneras en que ChucK se puede comunicar con otros programas, computadores y dispositivos de control. Brevemente cubrimos cómo MIDI puede ser usado con ChucK, usando un dispositivo externo MIDI (como un teclado) para tocar ChucK como un sintetizador, y cómo ChucK puede controlar otros sintetizadores, tanto en formatos software como hardware externo. También introducimos Open Sound Control (OSC), que es otra manera standard para comunicación de programas y dispositivos musicales. Luego observamos entradas y salidas por puerto serial, que nos permite conversar con aún más dispositivos.

Los apéndices cubren muchos detalles e incluyen más ejemplos de las características y capacidades de ChucK. El apéndice A cubre la instalación y puesta en marcha del entorno integrado de desarrollo (IDE, por Integrated Development Environment o Entorno de Desarrollo Integrado) miniAudicle, y el apéndice G cubre cómo usar ChucK en la línea de comandos (interfaz de texto).

El apéndice B documenta todas las funcioens de la biblioteca interna de ChucK. El apéndice C documenta todos los UGens incluidos. El apéndice G cubre ChucK en la línea de comandos. El apéndice D cubre OSC. El apéndice E cubre entrada y salida de archivos y el apéndice F cubre entrada y salida por puerto serial. El apéndice H elabora sobre formas de extender el idioma ChucK en sí mismo con nuevos UGens.

PARA AQUELLOS INTERESADOS EN LA SATISFACCIÓN INSTANTÁNEA

Si eres impaciente y quieres ver el poder de lo que ChucK es capaz de hacer (incluso sin entender exactamente cómo), una vez que hayas instalado el miniAudicle (apéndice A) y ejecutado tus dos primeros programas ("Hello World" y "Hello Sine", sección 1.2), puedes escribir y ejecutar el ejemplo de final de capítulo del listado 6.15. Inspirado por este poderoso programa de ChucK, puedes entonces volver atrás y sistemáticamente trabajar al ritmo del libro.

PARA LOS PROGRAMADORES FAMILIARIZADOS CON OTROS LENGUAJES

El apéndice A habla de cómo instalar y ejecutar el miniAudicle IDE (entorno de desarrollo integrado), y cómo instalar y correr ChucK desde la línea de comandos (Terminal, Command, etc.). Si no has instalado aún ChucK y/o miniAudicle en tu máquina, deberías empezar por el apéndice A.

La sección 1.2 ("Tus primeros programas en ChucK") muestra el miniAudicle y te introduce al operador ChucK y la manera que tiene ChucK de manejar el tiempo. Si eres alguien que prefiere la línea de comandos, consulta el apéndice G ("ChucK en la línea de comandos") para revisar las instrucciones.

¿CUÁL ES LA DIFERENCIA QUE CHUCK HACE?

En la superficie, ChucK se parece mucho a otros lenguajes como Java y Objective C, con unas pocas diferencias principales. La primera y más importante, el operador ChucK (=>) es usado para asignación, conexión de Unit Generators (UGens) de audio, entre otros. Diseñado para parecerse a una flecha indicando dirección, el operador ChucK promueve el flujo de conexiones de audio de izquierda a derecha, asignación y tiempo, entre otros. En Chuck el signo igual (=) no tiene utilidad, así que si estás familiarizado con un lenguaje que tiene una sintaxis como:

```
float temp = 0.0;
```

en ChucK tendrás que acostumbrarte a

```
0.0 => float temp;
```

Otra cosa de ChucK que probablemente te será nueva es cómo ChucK maneja el tiempo. Como programador tienes que explícitamente controlar el tiempo en tu código, así que te recomendamos que revises detalladamente la sección 1.4 ("Tiempo en ChucK: se trata de now"). Los tipos de datos time y duration son fundamentales para ChucK, son lo que lo hacen tan particular, y no serás capaz de programar con éxito (ni escuchar algún sonido) a menos que los aprendas.

Desde aquí, puedes motivarte con un par de ejemplos del poder de ChucK, escribiendo y ejecutando ejemplos, como los del listado 3.8 y/o del listado 6.15.

Deberías entonces volver atrás y trabajar con el capítulo 4 para aprender cómo ChucK maneja archivos de audio. Terminarás este capítulo con las

máquinas de ritmo (drum machines) del listado 4.11.

Si estás satisfecho con tener experiencia con los ejemplos hasta el momento, estás probablemente listo para saltar directamente a la parte 2 (empezando en el capítulo 6), y aprender todo sobre los poderosos UGens disponibles en ChucK. Desde este punto todo será nuevo, así que sigue leyendo y ejecutando los ejemplos. La siguiente cosa fundamentalmente nueva sobre ChucK que te encontrarás será cubierta en el capítulo 8 donde aprenderás sobre sporking (como forking) shreds (como threads, pero en ChucK). Objetos y clases (capítulo 8) pueden parecerte familiares, así como también los eventos (capítulo 10), pero ChucK es único ya que los eventos pueden ser generados por múltiples dispositivos externos (palance de mando, MIDI, Open Sound Control), así que te debería parecer distinto a lo que conoces. El capítulo 11 termina con MIDI, OSC y serial, para usuarios y programadores avanzados.

Existen también apéndices muy completos al final con referencias a prácticamente todo lo relacionado con ChucK, así que recuerda que puedes revisarlo, así como también el índice.

Sobre el código

Todo el código en este libro tiene una tipografía especial, que lo separa del texto circundante. En muchos listados, el código es comentado para destacar los conceptos clave, y a veces se usa un numerado de las líneas para proveer de más información sobre el código.

En el libro digital, algunos términos y texto y código aparecen de color, tal como los verías si lo escribieras en la ventana del editor miniAudicle. Esto se debe a que miniAudicle reconoce las palabras reservadas y las colorea

según su tipo.

Prácticamente todo el código mostrado en el libro puede ser encontrado en varios formas en el código fuente de ejemplo que acompaña este libro. El código de ejemplo puede ser descargado de forma gratuita desde el sitio web de Manning en www.manning.com/ProgrammingforMusiciansandDigitalArtists.

El código de ejemplo (incluyendo los archivos de audio relacionados) es instalado automáticamente cuando instalas ChucK. En un Mac puedes encontrarlo en /Library/ChucK/examples/book/digital-artists/, mientras que en Windows está ubicado en C:\Program Files/ChucK/examples/book/digital-artists. En Linux, si sigues el procedimiento de instalación del apéndice A, el código de ejemplo se encuentra en /usr/local/share/doc/chuck/examples/book/digital-artists/.

Todo el código de ejemplo de este libro puede ser accedido directamente desde el miniAudicle navegando a File > Open Example y encontrando book/digital-artists en el Example Browser.

NOTA: ChucK funcoina en Mac OS X 10.5+ o posterior, Windows XP o posterior, o un sistema Linux apropiado.

Author Online

La compra de Programación para músicos y artistas digitales incluye acceso gratuito a un foro privado en línea administrado por Manning Publications, donde puedes hacer comentarios sobre el libro, preguntar aspectos técnicos, y recibir ayuda de los autores y de otros usuarios. Para acceder este foro y subscribirte a él, dirígite a

www.manning.com/ProgrammingforMusiciansandDigitalArtists. Esta página brinda información sobre cómo acceder al foro una vez que te has registrado, qué tipo de ayuda está disponible y las reglas de conducta en el foro.

El compromiso de Manning hacia nuestros lectores es proveer un espacio donde se puede llevar a cabo un diálogo significativo entre lectores individuales y autores. No es un compromiso a una cantidad específica de participación de parte de los autores, cuyas contribuciones a AO continúan siendo voluntarias (y sin pago). Sugerimos que le preguntes a los autores cuestiones desafiantes para que no se desvíe su interés.

Acerca de los autores

AJAY KAPUR es actualmente el director del programa en tecnología musical (MTIID) en California Institute of the Arts, además de ser el decano asociado de investigación y desarollo en artes digitales. También es profesor titular de arte sonoro en New Zealand School of Music en Victoria University de Wellington. Ajay también es co-fundador (junto con Perry y otros) de Kadenze, un emprendimiento de educación en línea de artes y tecnología. Recibió un doctorado interdisciplinario en 2007 en University of Victoria, combinando ciencias de la computación, ingeniería eléctrica, ingeniería mecánica, música y psicología, enfoacdo en sistemas inteligentes de música y tecnología de medios. Ajay se graduó con una licenciatura en ciencias de la ingeniería y en ciencias de la computación de Princeton University en 2002.-

PERRY R. COOK recibió una licenciatura en artes (BA) en música en 1985

y una licenciatura en ciencias (BS) en ingeniería eléctrica de University of Missouri, Kansas City, graduándose con máximos honores (Magna Cum Laude). Recibió un master y un doctorado en ingeniería eléctrica en Stanford en 1990. Además de trabajar para compañías como NeXT Inc., Media Vision, Xenon/Chromatic e Interval Research, continuó en Stanford como director técnico del Center for Computer Research in Music and Acoustics, hasta que se hizo profesor en Princeton University en 1996, como profesor de ciencias de la computación, en conjunto con el departamento de música. Cook es también el autor de Synthesis Toolkit en C++ (STK), el que co-mantiene con Gary Scavone. Perry es también coautor (con Ge Wang) del lenguaje de programación de audio ChucK. Actualmente es profesor emérito en Princeton, y tiene pasantías como profesor y artista en California Institute of the Arts, Arizona State University, entre otras instituciones. Perry es también un consultor y consejero fundador del emprendimiento de música con tecnología móvil Smule, y cofundador (con Ajay entre otros) de Kadenze, un emprendimiento de educación en línea de artes y tecnología.

SPENCER SALAZAR es un estudiante de doctorado en el Stanford Center for Computer Research in Music and Acoustics (CCRMA), donde investiga distintas formas de performance y experiencia musicales basadas en computadores. Previamente creó interfaces de hardware y software nuevas para el lenguaje de programación de audio ChucK, desarrolló prototipos para electrónica de consumidor para compañías de tecnología de punta, hizo la arquitectura a gran escala para interacciones musicales sociales para la compañía de apllicaciones móviles Smule, compuso para conjuntos de computadores y teléfonos móviles, y enseñó muchos talleres en temas relacionados a la música por computador. Recibió su licenciatura en ciencias de la computación (BSE) de Princeton University en el año 2006.

GE WANG es profesor asistente en el Center for Computer Research in Music and Acoustics (CCRMA) de Stanford University. Recibió su licenciatura en ciencias de la computación (BS) en el año 2000 en Duke University y su doctorado en ciencias de la computación en el año 2008 de Princeton University (profesor guía Perry Cook). Ge es el creador (con Perry) y el arquitecto jefe del lenguaje de programación de audio ChucK. Es el director fundador de la Stanford Laptop Orchestra (SLOrk) y de la Stanford Mobile Phone Orchestra (MoPhO). Ge es el cofundador del emprendimiento de música para dispositivos móviles Smule, y el diseñador de las aplicaciones Ocarina y Magic Piano para iPhone, que cuentan con más de 80 millones de usuarios.