What's up ChucK? Development Update 2024

Marise van Zyl
CCRMA, Stanford University
Stanford, CA, United States
marise@ccrma.stanford.edu

Ge Wang
CCRMA, Stanford University
Stanford, CA, United States
ge@ccrma.stanford.edu

ABSTRACT
Since its inception in early 2000s, the ChucK music programming language has undergone many expansions and changes. In the early years, there was a flurry of contributions that are still in use today. During the 2010s, there was a notable decrease in ChucK development (despite a few dedicated individuals who kept the language on life support). Recently, however, ChucK development has experienced something of a resurrection. This paper highlights the major initiatives since 2018, including new core language features, ChuGL (graphics), ChAI (AI), Chunity (ChucK in Unity), Chunreal (ChucK in Unreal Engine), WebChucK (ChucK in browsers), and further extensions to the language through Chugins (ChucK plugins). Furthermore, we will highlight future directions of ChucK development.

Author Keywords
ChucK, programming language, computer music software development.

CCS Concepts
• **Applied computing** → **Sound and music computing**;• **Software and its engineering** → *Domain specific language;* • **Human-centered computing**→ *Systems and tools for interaction design;*

1. EXPANSION AND STASIS
ChucK is a strongly-timed music programming language [1] originated by Ge Wang and Perry Cook at Princeton University. First released in 2004 under the GPL open-source software license, the defining features of the language include a deterministic, time-based, concurrent programming model and an on-the-fly programming workflow. ChucK development saw great expansion, as researchers contributed many features to the language and ecosystem [2, 3, 4, 5, 6], which have been used in laptop orchestras [7,8,9], classrooms [10], and in popular mobile music apps [11].

In the subsequent decade (circa 2008-2018), ChucK development experienced a drastic slow-down due to various factors including Wang joining the faculty at Stanford University and the founding of *Smule*, a mobile music startup composed of several of ChucK's core developers. While early *Smule* apps and similar research initiatives used ChucK [11, 12, 13, 14], the projects focused on application rather than language development. Nevertheless, a few dedicated individuals kept the language alive, although on "life support" [15, 16].

2. RESURRECTION
Since 2018, ChucK has experienced a development renaissance that has resulted in a proliferation of additions and enhancements to the language, workflow, and community. It is difficult to pinpoint exactly when this renewal began.

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). Copyright remains with the author(s).

NIME'24, September 4–6, 2024, Utrecht, The Netherlands

But one might look to the creation of Chunity (ChucK in Unity) and WebChucK by Jack Atherton in 2017 and 2018. Another catalyst might be the publication of *Artful Design* [17] which highlighted ChucK as a tool for the kind of aesthetics-driven design the book argues for. On a practical note, the completion of the book also allowed Wang to return to language development on ChucK. A third development that solidified momentum for the ChucK development renewal was Terry Feng's creation of WebChucK IDE in Chris Chafe's Fundamentals of Computer-generated Music course at CCRMA. WebChucK IDE expanded access and allowed ChucK to be both run and coded wherever there is internet access. Meanwhile, Celeste Betancur joined the CCRMA Ph.D. program and elevated, by example, what ChucK was capable of as a modern musical programming language for both academic works and as a battle-tested live-coding tool in popular club settings. Finally, curricular demands led to the creation of ChAI for Wang's new "Music and AI " critical-making course, and, later, to ChuGL for the "Music, Computing, Design" course at CCRMA. These developments made ChucK feel relevant again and organically reignited enthusiasm in ChucK development.

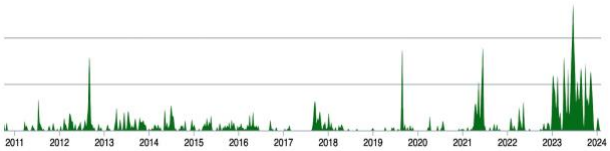

Figure 1. GitHub commits to the ChucK core repository from 2010 until February 2024.

2.1 ChucK Kitchen Cabinet (CKC)
The "ChucK Kitchen Cabinet" is a group of ChucK developers at CCRMA (and remotely) that form the core of ChucK development. The group consists of a dozen members who meet regularly, plan the development roadmap, work on various aspects of language development, and organize sprints and development "hackathons" It is unclear exactly how the group came to be, but by the fall of 2022, a few stragglers organically found themselves in the same small room in CCRMA, working on ChucK. Over time, more joined and the sense of community and commitment towards a common goal solidified the working group. "OG" ChucK developers such as Perry Cook and Spencer Salazar contribute to the CKC as remote members.

A version of the colloquial "kitchen cabinet", the CKC is a group of individuals each with their skills and interests to bring to bear on various aspects of the language. In this setting, there is both structure and freedom for all the members, providing a balance of function and fun. This is perhaps the closest ChucK development has come to a formal and centralized development process and is certainly the largest by group size.

Figure 2. Members of the ChucK Kitchen Cabinet (CKC) at a development "hackathon".

3. NEW DEVELOPMENTS

3.1 Chunity: ChucK in Unity

Chunity (ChucK for Unity) is a programming environment for the design of interactive audiovisual games, instruments, and experiences created by Jack Atherton [18]. It embodies an audio-driven, sound-first approach that integrates audio programming and graphics programming in the same workflow, taking advantage of strongly-timed audio programming features of the ChucK programming language and the real-time graphics engine found in Unity. In short, Chunity enables one to program ChucK inside the Unity game development framework and provides mechanisms for communication between ChucK and C# / Unity. At CCRMA, Chunity has been used as a teaching tool as well as a research tool in CCRMA's VR Design Lab; it has been used in projects such as VVRMA [19], RayTone [20], and sVoRk. (https://chuck.stanford.edu/chunity)

Figure 3. An audiovisual cityscape made with Chunity.

3.2 WebChucK + IDE

WebChucK allows ChucK to be used in a browser [21, 22]; its development and use has drastically grown since Jack Atherton's initial work on WebChucK in 2018. WebChucK works by compiling ChucK's source code (in C++) to WebAssembly using Emscripten and runs via the AudioWorkletNode interface of the Web Audio API. WebChucK works on modern browsers, including on mobile phones and tablets. Its modular nature makes it possible to embed real-time interactive audio synthesis into any website, which have been used for installation, shareable musical instruments, research, and teaching. A recent development for WebChucK is the WebChucK IDE, developed by Terry Feng in 2022 [23]. The IDE is an online programming sandbox for ChucK which allows for programming straight in the browser without downloading any software.
(https://chuck.stanford.edu/webchuck | https://chuck.stanford.edu/ide)

3.3 ChAI: Interactive AI Tools in ChucK

Despite its best efforts, ChucK could not escape AI and thus ChAI (ChucK + AI) development commenced in 2022, led by Ge Wang and Yikai Li. ChAI contains a set of tools, algorithms, datasets, examples for working with supervised learning, unsupervised learning, neural networks, interactive music generation, and education [24]. ChAI was used as a primary tool in a Music and AI class taught at CCRMA by Ge Wang for the first time in the winter of 2023 and again in 2024. Objects include MLP (multilayer perceptron), KNN, SVM, HMM, PCA, Word2Vec, and a full port of Wekinator [25]. Additional audio features have been added to the unit analyzer framework, including MFCC, Chroma, Kurtosis, SFM. Despite "caving" to the AI trend, ChAI was not designed as a generative tool but instead prioritizes "human-in-the-loop" interactive AI and an ethos of critical-making (critical thinking + creative making). Similarly, Wang's Music and AI course stresses the need to be thoughtful about the implications of what we make when designing anything with AI, and to be cognizant of its power to shape not only behavior and workflows, but the very culture in which we live, work, and play [26].

3.4 ChuGL: Unified Graphics and Sound Programming in ChucK

Over the years, several attempts have been made to bring high-performance 3D/2D graphics programming to ChucK, notably by Philip Davidson in 2003 and by Spencer Salazar in 2013. Following this ten-year trend, Andrew Zhu Aday and Ge Wang designed ChuGL as a unified audiovisual programming framework built directly into the ChucK language. This latest effort introduces an entirely new, retained-mode, graphics engine architecture that works performantly with real-time audio synthesis, while introducing a new language construct called Graphics Generators (GGens), analogous to Unit Generators (UGens). Building on past iterations, this latest ChuGL successfully combines ChucK's real-time audio synthesis capabilities with a hardware-accelerated 3D graphics engine into a single strongly-timed language [27]. Presently, ChuGL now ships as part of the standard language distribution and is used as the primary tool in "Music, Computing, Design" course at Stanford University.
(https://chuck.stanford.edu/chugl)

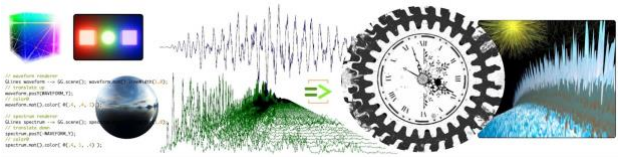

Figure 4. Unified, audiovisual programming in ChucK with ChuGL.

3.5 Core Language Features

In addition to the above expansions to the ChucK ecosystem, much work has gone into expanding the core language with new features and quality-of-life improvements. These include object constructors (finally!), operator overloading, improved memory management, additional control structures including for-each, advanced Chugins programming API, CKDoc (a dynamic in-engine documentation generator), and numerous optimizations and bug fixes (and likely new bugs). Furthermore, audio driver selection has been added on all platforms (macOS, Linux, Windows), including ASIO and WASAPI on windows. Support for Apple Silicon has been added as well, with profound (3x or more) performance improvements across the board.

3.6 Documentation

Many of the important changes and updates to the language are less obvious, yet perhaps more salient. In the early part of 2023, ChucK underwent a massive documentation overhaul with a long-overdue API documentation of standard ChucK objects, each with example code. (https://chuck.stanford.edu/doc/reference)

This documentation replaces the previous static (and incomplete and out-of-date) documentation and is now generated dynamically (using CKDoc) from within the ChucK runtime type system to ensure accuracy with respect to the latest actual language API. The ChucK webpage has been updated with new content (e.g., examples, video tutorials by ChucK community member Clint Hoagland, links to resources, WebChucK IDE), as well as a new cosmetic update.

3.7 Chugins

A Chugin, in short, is a plugin for ChucK. Each Chugin operates as a distributable dynamic library, typically written in C or C++ compiled to native machine code, which ChucK can load and use at runtime as expansion to the language with new audio synthesis and analysis functionalities, as well as general purpose programming libraries. While chugins support has existed since 2012 [15], the past few years has seen many new Chugins (ConvRev, FluidSynth, NHHall, LADSPA, Hydra, RAVE: IRCAM's variational autoencoder for real-time audio synthesis). Moreover, the Chugins API has been expanded to give Chugin creators greater power and access into the ChucK virtual machine (VM) runtime. In fact, ChuGL runs as a special Chugin that can natively synchronize graphics with the ChucK type system and VM and access a well-defined set of API functions.

3.8 Additional ChucK Integrations

ChucK has been increasingly integrated with and into other software systems. In addition to Chunity, Eito Murakami developed Chunreal, an integration of ChucK into Unreal Engine 5 that allows users to compile and run ChucK code at runtime in communication with UE5 objects. Development is ongoing. Another integration is FaucK, which combines the powerful, succinct Functional AUdio STream (FAUST) language with the strongly-timed ChucK audio programming language [28] FaucK allows programmers to on-the-fly evaluate Faust code directly from ChucK code and control Faust signal processors using ChucK's sample-precise timing and concurrency mechanisms. Further developments include ChucK Designer (ChucK in *TouchDesigner*; David Braun), pd-chuck (ChucK in *PureData*), and ChucK-Max (ChucK in *Max MSP*) both by shakfu.

4. CONCLUSION AND FUTURE WORK

Since ChucK's initial release in 2004, the ChucK community has been an integral part of its being. The community not only brings together programmers using ChucK for a variety of purposes, but the community also contributes to and ultimately pushes development forward in the form of feedback, shared work, code, feature requests, bug reports, pull requests, Chugins, and entirely new ways of using ChucK. Today, ChucK Community exists as a public Chuck Community Discord Server (https://discord.gg/Np5Z7ReesD), mailing lists, and a Facebook group.

2024 marked the 20th anniversary of ChucK and development is actively progressing on all aforementioned projects, as well as new initiatives including a ChucK Package Manager and a Stanford VR Orchestra powered by Chunity. Many factors have contributed to the resurrection of ChucK. For reasons unknown, individuals are choosing to use their time to work on a computer music programming language. While this endeavor might not save the world, it does hold some intrinsic value to the people doing it; and isn't that reason enough?

5. ACKNOWLEDGMENTS

Firstly, thank everyone in the ChucK Kitchen Cabinet and ChucK Community, including Nick Shaheed, Terry Feng, Celeste Betancur, Alex Han, Yikai Li, Andrew Zhu Aday, Eito Murakami, Spencer Salazar, Perry Cook, Kunwoo Kim, Clint Hoagland, David Braun, Max Jardetzky, Mike Mulshine, Rob Hamilton, Jack Atherton, Rebecca Fiebrink, shakfu, Dana Batali, Michael Heuer, Ajay Kapur, Chris Chafe, students of courses at Stanford University, Princeton University, and CalArts.

6. ETHICAL STANDARDS

This paper complies with the NIME ethical standards. No human or animal participants are involved.

For more information visit the ChucK website:
https://chuck.stanford.edu/

7. REFERENCES

[1] Wang, G., P. R., Cook, and S. Salazar. 2015. "ChucK: A Strongly Timed Computer Music Language" *Computer Music Journal 39*:4. doi:10.1162/COMJ_a_00324.

[2] Wang, G., P. R. Cook. 2004. "On-the-fly Programming: Using Code as an Expressive Musical Instrument.". *In Proceedings of the International Conference on New Interfaces for Musical Expression*. Hamamatsu, Japan.

[3] Wang, G., A. Misra, A. Kapur, and P. R. Cook. 2005. "Yeah ChucK It! => Dynamic Controllable Interface Mapping.". *In Proceedings of the International Conference on New Interfaces for Musical Expression*. Vancouver.

[4] Salazar, S., G. Wang, and P. R. Cook. 2006. "miniAudicle and ChucK Shell: New Interfaces for ChucK Development and Performance." *In Proceedings of the International Computer Music Conference.* New Orleans.

[5] Wang, G., R., Fiebrink, and P. R. Cook. 2007. "Combining Analysis and Synthesis in the ChucK Programming Language." *In Proceedings of the International Computer Music Conference*. Copenhagen.

[6] Fiebrink, R., G. Wang, and P. R. Cook. 2008. "Foundations for On-the-fly Learning in the ChucK Programming Language." *In Proceedings of the International Computer Music Conference*." Belfast, Ireland.

[7] Trueman, D., P. R. Cook, S. Smallwood, and G. Wang. 2006. "PLOrk: Princeton Laptop Orchestra, Year 1." *In Proceedings of the International Computer Music Conference*. New Orleans.

[8] Smallwood, S., D. Trueman, P. R. Cook, and G. Wang. 2008. "Composing for Laptop Orchestra." *Computer Music Journal. 32*(1):9-25.

[9] Wang, G., N. Bryan, J. Oh, and R. Hamilton. 2009. "Stanford Laptop Orchestra (SLOrk)." *In Proceedings of the International Computer Music Conference.* Montreal.

[10] Wang, G., D. Trueman, S. Smallwood, and P. R. Cook. 2008. "The Laptop Orchestra as Classroom." *Computer Music Journal. 32*(1):26-37.

[11] Wang, G. 2014. "Ocarina: Designing the iPhone's Magic Flute." *Computer Music Journal. 38*(2):8-21.

[12] Fiebrink, R., G. Wang, and P. R. Cook. 2008. "Support for MIR Prototyping and Real-time Applications of the ChucK Programming Language." *In Proceedings of the International Conference on Music Information Retrieval.* Philadelphia.

[13] Fiebrink, R. A. 2011. Real-time human interaction with supervised learning algorithms for music composition and performance. *PhD Thesis*. Princeton University.

[14] Wang, G., S. Salazar, J. Oh, and R. Hamilton. 2015. "World Stage: Crowdsourcing Paradigm for Expressive Social Mobile Music." *Journal of New Music Research. 44*(2):112-128.

[15] Salazar, S. and G. Wang. 2012. "Chugens, Chubgraphcs, and Chugins: 3 Tiers for Extending ChucK." *In Proceedings of the International Computer Music Conference.* Slovenia.

[16] Kapur, A., P. R. Cook, S. Salazar, and G. Wang. 2015. *Programming for Musicians and Digital Artists: Creating Music with ChucK*. Manning Press. (ISBN: 978-1617291708).

[17] Wang, G. 2018. *Artful Design: Technology in Search of the Sublime (a MusiComic Manifesto*). Stanford University Press. (ISBN: 978-1503600522.

[18] Atherton, J. and G. Wang. 2018. "Chunity: Integrated Audiovisual Programming in Unity.*" In proceedings of New Interfaces for Musical Expression.*

[19] Kim. K. and G. Wang. 2024. "VVRMA: VR Field Trip to a Computer Music Center." *New Interfaces for Musical Expression*. Utrecht.

[20] Murakami, E., Burnett, J., G. Wang. 2024. "RayTone: A Node-based Audiovisual Sequencing Environment." *New Interfaces for Musical Expression*. Utrecht.

[21] Chafe, C., Ge Wang, M. R. Mulshine, J. Atherton. 2023. "What Would a Webchuck Chuck?" *The Journal of the Acoustical Society of America. 153*(3) Supplement A35. https://doi.org/10.1121/10.0018058.

[22] Mulshine, M. R., G. Wang, J. Atherton, C. Chafe, T. Feng, C. Betancur. 2023. "WebChucK: Computer Music Programming on the Web.*" In proceedings of New Interfaces for Musical Expression.* Mexico City.

[23] Feng, T., C. Betancur, M. R. Mulshine, C. Chafe, G. Wang. 2023. "WebChucK IDE: A Web-Based Programming Sandbox for ChucK." *In proceedins of Sound and Music Computing.* Stockholm.

[24] Li, Y, Wang, G. 2024. "ChAI: Interactive AI Tools in ChucK." *New Interfaces for Musical Expression.* Utrecht.

[25] Fiebrink, R., & Cook, P. R. 2010. "The Wekinator: a system for real-time, interactive machine learning in music." *In Proceedings of The Eleventh International Society for Music Information Retrieval Conference*. (Vol. 3, pp. 2-1).

[26] Wang, G. 2019 "Humans in the Loop: The Design of Interactive AI Systems." *Stanford Human-Centered Artificial Intelligence*.

[27] Aday, A.Z., Wang, G. 2024. "ChuGL: Unified Audiovisual Programming in ChucK." *New Interfaces for Musical Expression.*Utrecht.

[28] Wang, G. and R. Michon. 2016. "FaucK!! Hybridizing the FAUST and ChucK Audio Programming Languages." *Sound and Music Computing.*