Proceedings of the Spatial Audio Gathering (SAG26)
Leeds Beckett University, Leeds, UK. March 31–April 1, 2026

ChucK'd Into Space With Chumbi, Higher-Order Ambisonics in the ChucK Programming Language

Everett M. Carpenter
Rensselaer Polytechnic Institute
Department of Arts
Troy, NY
carpee2@rpi.edu

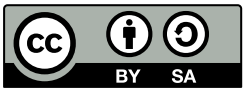
Abstract

This paper introduces Chumbi, a collection of ambisonic audio processors and utility classes for the ChucK programming language. Chumbi contains an ambisonic encoder, various decoders, a spherical harmonic calculator, and soundfield rotation utilities, all of which are written in C++. It has been created as a new paradigm for spatial audio in computer music, focussing on the design of musical interactions and instruments as well as experimentation in the recontextualization of ambisonic methods. Chumbi can be accessed as a package via ChucK's package manager ChuMP or the author's site. Chumbi aims to equip composers and artists with the tools for designing spatial-minded digital instruments.

Keywords: Ambisonics; HOA; Design; Computer Music;

1 Introduction

Existing technology for spatial audio prioritizes the delivery of ready-made processors and devices over low-level, exploratory alternatives. Software such as Virtual Studio Technology (VST) prevents users from reviewing and editing underlying signal processing algorithms, leading to most users adopting a "black-box" approach to spatial audio. The ChucK programming language provides the necessary tools to construct real-time, on-the-fly code-based creation (Wang et al., 2015). ChucK enables artists and composers to consider the design of their instruments and tools as a creative process rather than one of necessity. Previously, ChucK lacked a robust API for ambisonic processing, only offering a limited ambisonic encoder. Chumbi, a collection of unit generators (UGens) was introduced as a solution to the lack of ambisonic technology in ChucK all the while still maintaining the ChucK design philosophy of design as an artful process. Chumbi is not simply a collection of UGens providing ambisonic processing, but is a paradigm for the design of spatially-aware instruments, environments and tools. This paper outlines the existing paradigms and tools for ambisonic processing, Chumbi's design and architecture, ending with projects using Chumbi as well as discussion regarding future work.



This work is licensed under a Creative Commons "Attribution-ShareAlike 4.0 International" licence.

2 Related Work

There is a large collection of tools for ambisonic processing already available, including IRCAM's *Spat*[1], IEM's plugin suite[2] and Joseph Anderson's *Ambisonic Toolkit* (*ATK*)[3], all of which embody a distinct paradigm. In particular, *Spat* orients itself around GUI based control of source positioning with additional controls for perceptual features such as "warmth", "presence" and "brilliance". The plugin suite actively developed by the IEM focusses on handling intricate signal processing internally and providing processors of varying complexity. Due to this approach, the IEM's suite has appealed to many users of varying experience. The *ATK* approaches the use of ambisonics from a modular lens, providing various processors which can be combined and arranged in multiple ways. However, the *ATK*'s presented paradigm is strictly focused on three stages, encoding, transforming and decoding, with less explicit consideration to the recontextualization of ambisonic theory.

Before Chumbi's official release, ChucK provided the aforementioned third order ambisonic encoder, `AmbiPan3`. Other solutions included the early developments of what would become Chumbi, which consisted of multidimensional arrays of `Gain` UGens used to apply weightings to multichannel audio streams. Due to these objects being written within the ChucK language, composers and performers not only had to manage the application of ambisonics, but the operation of it as well. This system proved to be too tedious and convoluted for real-time use. Additionally, this paradigm was computationally expensive, with spherical harmonics being calculated by ChucK, rather than a lower level process. Thus, the need existed for a readily available framework that could provide ChucK users with a real-time ambisonic processing suite which was both flexible and efficient.

3 Design and Architecture of Chumbi

An effective and capable toolkit regarding any specialized topic or process provides a means for straightforward use while also presenting the tools for experimental play. Chumbi's design embodies this design philosophy by providing varying degrees of control, while still opening paths for hacking and recontextualization of its members. By offering varying levels of control, Chumbi can be utilized in pedagogical applications as well as research and development of spatial sound technology.

Chumbi is implemented as a set of ChuGins which are pre-compiled dynamic linking libraries written in C++, an API introduced by Salazar and Wang (2012). ChuGins are loaded at runtime by the ChucK virtual machine, which executes all ChucK scripts. ChucK's package manager, ChuMP, provides all of Chumbi's components from first to fifth order. For the reader interested in orders higher than fifth, Chumbi is open source[4]. As a companion to Chumbi, a spherical harmonic projection calculator is now built into the language's math library. Spherical harmonics in ChucK follow the SN3D weighting scheme in an attempt to formalize the standard set out by Nachbar et al. (2011). However, this does not mean users can create or alter B-format audio outside of the SN3D standard, as Chumbi's UGens feature a user-defined weighting vector.

All encoders and decoders contained within Chumbi are designed to operate on spatial information via

[1] `https://forum.ircam.fr/projects/detail/spat/`
[2] `https://plugins.iem.at/`
[3] `https://www.ambisonictoolkit.net/`
[4] `https://github.com/everettmcarpenter/Encode-Decode.chug`

reduced spherical coordinates (azimuth and elevation in accordance with the ambix format), or spherical harmonic projections allowing users to provide a vector whose components are interpreted as projection coefficients upon the spherical harmonic basis functions. These two approaches to control are intended to enable the user in their creative application of ambisonic technology. Whether they intend to spatialize pre-recorded material via virtual source panning, or embed ambisonic processing as a key component of audio synthesis processes such as the recent work of Tondock and von Coler (2023). Once again, the design of Chumbi is with the intention of unmasking "black-box" technology at varying degrees so that users may delegate their desired amount of complexity to the software itself.

4 Applications of Chumbi

The following projects utilize Chumbi's components to achieve real-time ambisonic processing with varying levels of control.

4.1 Use of Chumbi in Digital Instruments

AmbiGranny is an ambisonic granulator which positions sonic grains within space according to a *spread* parameter. Users control AmbiGranny via a laptop or external keyboard, which they can use to change pitch, length, positioning and spread of grains. AmbiGranny was designed as an instrument which explores the entanglement of spatiality and temporality, evolving sonic objects over time and physical space.

As mentioned previously, Chumbi's design intends to accommodate both old and new ways of using ambisonic audio. Investigations in recontextualizing ambisonic methods led to experiments in the spherical harmonic coding of timbre, placing parameters and features of sound upon the sphere. Similarly, spectral data was encoded for resynthesis methods, with both the former and the latter being controlled by physical controllers and stochastic processes. Using these methods, a spherical coding of timbre was introduced and used in performance applications where users navigated a virtual sphere of sound. As the sphere was traversed physical positioning resulted in an interpolated timbre, associating spaces with sound qualities.

4.2 Audiovisual and Multimedia Applications

Using ChucK's audiovisual framework ChuGL created by Zhu and Wang (2024), a visualization program was created in ChuGL which displays the effects of ambisonic order reduction on localization accuracy. Users are presented with a wireframe sphere containing a vector whose components are calculated via Gerzon's localization criteria (Gerzon, 1992). The Gerzonian Vector Visualizer not only provides an intuitive interface to observe localization accuracy across various ambisonic orders, but it may also serve as a tool for evaluating the performance of ambisonic decoders.

As of 2018, the Unity game engine is capable of running ChucK instances (Atherton and Wang, 2018), allowing users to mix audio and visual programming using Chunity, a package available on the Unity Asset Store. Chunity is capable of loading and running ChuGins, enabling users to implement ambisonic processing within their Unity projects via Chumbi. Various implementations have been tested, including a scene where users can leverage spatial positioning of themselves while performing together.

Developed alongside Chumbi is the multimedia project titled "Sonically Immersive Tactile Environments" (SITEs), an immersive virtual reality (VR) experience utilizing haptics and real-time synthesis. Various

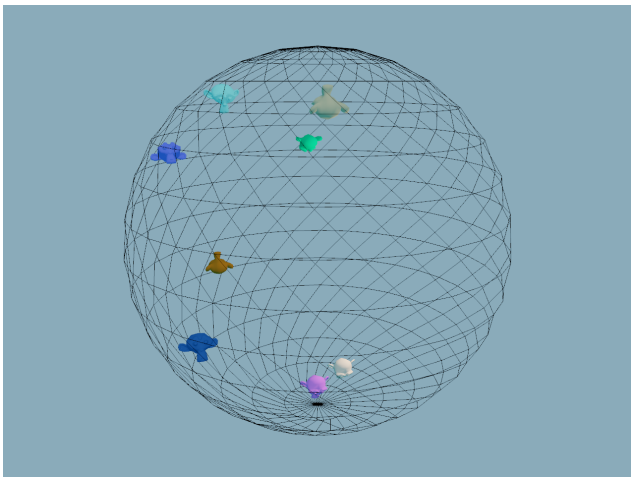


Figure 1: A spatial visualizer made with Chumbi.

prototypes of SITEs have been developed using Chumbi, creating subclasses of Chumbi's components to replicate certain types of spatial audio sources such as clouds, point sources, and stochastically scattered sounds. SITEs is an actively developed project with Chumbi becoming an increasingly more integrated element of its design.

5 Discussion and Future Work

Chumbi has enabled both the original use of ambisonics as well as the recontextualization of the theory for real-time audio programs, including bridges between spatial and timbral synthesis, spatial controllers within virtual reality, and pedagogical tools. This wide array of applications follows Chumbi's intended purpose, enabling users with varying needs to utilize it. However, Chumbi still lacks in what other ambisonic packages provide. Chumbi's active release does not support orders higher than fifth; it does not contain the already researched soundfield transformations of Kronlachner (2013); lastly, it does not provide an interchangeable format for decoder specification, preventing the distribution of ambisonic decoders. These features are to be developed for Chumbi, with the intention of the package being an all encompassing toolkit for users of all skill levels.

Acknowledgements

The author would like to extend thanks to Dr. Rob Hamilton, Dr. Kate Galloway, and Todd Voss of EMPAC. Additional recognition and appreciation is extended to Dr. Will Gibbons and the school of Humanities, Arts and Social Sciences under Rensselaer Polytechnic Institute for partial funding for travel to the Spatial Audio Gathering. Lastly, to all who tested, reviewed and listened to Chumbi.

References

Atherton, J. and Wang, G. (2018). Chunity: Integrated audiovisual programming in unity. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 102–107, Blacksburg,

Virginia, USA. Virginia Tech.

Gerzon, M. (1992). General metatheory of auditory localisation. *Journal of The Audio Engineering Society*, (3306).

Kronlachner, M. (2013). Spatial transformations for the alteration of ambisonic recordings. Master's thesis, Institut für Elektronische Musik und Akustik.

Nachbar, C., Zotter, F., Deleflie, E., and Sontacchi, A. (2011). Ambix -a suggested ambisonics format. *Ambisonics Symposium 2011*.

Salazar, S. and Wang, G. (2012). Chugens, chubgraphs, chugins: 3 tiers for extending chuck.

Tondock, H. and von Coler, H. (2023). A virtual instrumentfor ifft-based additive synthesis in the ambisonics domain.

Wang, G., Cook, P. R., and Salazar, S. (2015). Chuck: A strongly timed computer music language. *Computer Music Journal*, 39(4):10–29.

Zhu, A. and Wang, G. (2024). Chugl: Unified audiovisual programming in chuck. pages 351–358.