

Chumbi – A Toolkit For Accessing Ambisonics at Various Levels in ChuckK

Everett M. Carpenter

Rensselaer Polytechnic Institute, Troy, NY
carpee2@rpi.edu

ABSTRACT

While basic ambisonic processing has been possible in the ChuckK programming language, these solutions have been inefficient and hard to assemble, and required a significant amount of technical understanding before basic ambisonic output could be generated. Such complexity has been less than ideal for new or less technical users simply trying to apply ambisonic spatialization to their artistic work.

This paper introduces Chumbi, a collection of pre-compiled ambisonic audio processors and utility classes for the ChuckK programming language. Chumbi contains an ambisonic encoder, various decoders, a spherical harmonic calculator, and soundfield rotation utilities, all written in C++. Chumbi can be accessed as a package via ChuckK's package manager ChuMP, providing examples, documentation and all UGens detailed in this paper.

1. INTRODUCTION

Multiple methods of sound spatialization via loudspeaker technology have been developed to provide listeners with the immersion and auditory stimuli commonly found in sound fields [1, 2, 3]. Various methods of auditory localization such as interaural time and level differences (ITD and ILD), allow humans to identify the location from which a sound originates from [4]. Ambisonics is the process of segmenting a virtual or real sound field into a discrete set of angular components in an effort to replicate these cues. The practice of Higher Order ambisonics (HOA) looks to segment the sound field into large sets of components. Through higher degrees of segmentation, the angular resolution of the set becomes finer. Additionally, issues such as unrealistic virtual sources can be addressed with HOA, employing high resolution ambisonic reverberation algorithms [5, 6]. Angular beamforming is also possible with HOA, allowing for the derivation of monophonic signals which represent precise angular segments of the spherical sound field [7, 8, 9].

The ChuckK programming language [10] is utilized by musicians, composers and artists as a tool to create sound. ChuckK is a text based language, with a precise control over time. Historically, it has not maintained any dedicated ambisonic audio processors other than a third-order ambisonic encoder titled AmbPan3. In previous research, ambisonic

processing was accomplished with large arrays of gain altering unit generators (UGens). This did not require any external components from users, and could be done entirely within ChuckK. While this method was feasible, it was computationally inefficient and difficult for the new user to approach.

Chumbi is a package of UGens for the ChuckK programming language and contains a math library which allows users to easily interact with HOA on various levels of involvement. Chumbi was designed to invite engagement with users of varying experience in HOA. While most may prefer to operate at a higher level, users who are looking to understand, or already understand the underlying mathematics and signal processing of ambisonics may also engage with HOA at a very low level. In the first section, a brief preliminary will be given on notation used in ambisonics, and the signal flow of ambisonics. In the second section, an outline of Chumbi's components will be given. Finally in the third section, examples of Chumbi being used will be presented.

2. AMBISONIC THEORY

Ambisonic research defines a method of sampling the sphere via *spherical harmonics*, which produces lobes representing space within and on the surface of the sphere¹. While the acoustic and mathematical details of ambisonics are outside the scope of this paper, the Fourier-Bessel representation of the Helmholtz equation is the source of the spherical harmonic framework used in ambisonics [1, 2, 12, 13].

$$p(\vec{r}) = \sum_{n=0}^{\infty} r^n j_n(kr) \sum_{m=-n}^n B_n^m Y_n^m(\theta, \phi) \quad (1)$$

The real-valued spherical harmonic is described by Y_n^m where m is referred to as the degree and n is the order [14]. We refer to the collected spherical harmonics of a given order N as $\mathbf{y}_N(\Theta)$, where Θ represents a given (θ, ϕ) . Indices n and m fall within the boundary conditions $0 \leq n \leq N$ and $-n \leq m \leq n$ so that $\mathbf{y}_N(\Theta)$ takes the form:

$$\mathbf{y}_N(\Theta) = [Y_0^0(\Theta), Y_1^{-1}(\Theta), \dots, Y_N^N(\Theta)]^T \quad (2)$$

An object's angular position can be described by a vector of spherical harmonic functions. This notational framework is adopted from [2] in an effort to maintain consistent language within the research field.

Copyright: ©2026 Everett M. Carpenter et al. This is an open-access article distributed under the terms of the [Creative Commons Attribution License 3.0 Unported](https://creativecommons.org/licenses/by/3.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

¹ Poletti also defined a horizontal only approach in [11].

2.1 Signals in Ambisonics

Ambisonics transmits the sound field to be described through *encoded* signals. These sets of signals are referred to as B-Format signals, which are indexed by an ambisonic channel number (ACN). It is generally agreed that ACN is the expected ordering of ambisonic components [15]. Each ACN represents a spherical harmonic of some order, with respect to the origin point of the spherical sound field. By weighting and summing these signals, we can represent the acoustic properties of the spherical sound field. The typical workflow to achieve this is described in the rest of this section.

2.2 Ambisonic Encoding

B-Format signals are described by some order N , exactly as the spherical harmonics are in (2). B-Format signals contain $(N + 1)^2$ concurrent channels of audio to represent 3D sound fields, and can be truncated to $(2N + 1)$ for 2D sound fields². Throughout this paper, B-Format signals will be denoted by some vector \mathbf{b}_N . Therefore, unless otherwise stated, all \mathbf{b}_N take the form of:

$$\mathbf{b}_N = [B_0^0, B_1^{-1}, B_1^0, B_1^1, \dots, B_N^{N-1}]^T \quad (3)$$

The only difference between \mathbf{b}_N and \mathbf{y}_N is that \mathbf{b}_N is a product of \mathbf{y}_N and a scalar S . When applying ambisonics to digital signal processing, S takes the form of a discrete, real valued signal.

2.3 Ambisonic Decoding

Once B-Format signals are received, synthesized or recorded, they can be played back on L number of loudspeakers or rendered binaurally. Using loudspeakers for reproduction at order N , $(N + 1)^2$ loudspeakers are required for 3D playback, and $(2N + 1)$ for 2D playback. This is done by obtaining coefficients described in a square matrix, where each row represents coefficients for a loudspeaker. Having obtained \mathbf{D}_N , a column vector \mathbf{p}_N can be calculated, containing signals which correspond to a specific loudspeaker defined by each row of \mathbf{D}_N . This calculation is accomplished via the matrix-vector multiplication:

$$\mathbf{p}_N = \mathbf{D}_N \mathbf{b}_N \quad (4)$$

It is important to note that the matrix \mathbf{D}_N is where ambisonics begins to vary. The following section will define a few methods of obtaining \mathbf{D}_N .

3. COMPONENTS OF CHUMBI

Chumbi's UGens are written in C++ and loaded at runtime by the ChucK virtual machine. Created alongside Chumbi's UGens, a library meant for on-the-fly real valued spherical harmonic calculation is provided. Chumbi's pre-compiled package supplied by ChuMP supports up to 5th order ambisonics. Under the hood, Chumbi supports up to the 12th order ambisonics, but any order higher than 5th must be compiled by the user themselves. All UGens

² As described in [1], spherical harmonics of which $m = n$ are only dependent upon the azimuth θ , thus allowing for the removal of harmonics that depend on the zenith angle.

Figure 1. Instantiation of Chumbi's UGens

```
// first order ambisonic encoder...
Encode1 encoder;
// ambisonic decoder
Decode1 decoder;
// sine osc at 432.5Hz
SinOsc sinusoid(432.5);
// into the ambisonic domain!
sinusoid => encoder => decoder => dac;
```

in Chumbi have an inheritance hierarchy which provides base functions that can be called by all of Chumbi's components. The most important of these is the ability for all UGens to receive either simple spherical coordinates, using azimuth and zenith, or to receive spherical harmonics. This allows the user to design alternate weightings of spherical harmonics if desired. Instantiation of Chumbi is similar to other UGens in ChucK, but the order N of some UGen is defined. By defining the order at which a signal processor will operate, all required input and output buffers are allocated. This is shown in Fig.1

3.1 Ambisonic Encoding with *EncodeN*

EncodeN functions as Chumbi's ambisonic virtual source encoder. By implementing the input methods mentioned in Section 3, users can operate EncodeN with just two input values. This enables control structures such as physical interfaces, wireless interfaces and others to be easily attached to a ChucK shred which continuously feeds positional data to EncodeN.

3.2 A-Format \Leftrightarrow B-Format Conversion with *ABFormat* and *BAFormat*

In [4, 16], Gerzon began to define the mathematical theory behind ambisonic microphones. This has been expanded incredibly by further research such as [17] and [18, 19]. Chumbi contains two UGens which convert from A-Format to B-Format and vice versa. These UGens only operate at the first order due to HOA microphones often-times varying in design geometry. Similar to the workflow of other UGens in Chumbi, all that is required from the user is to patch their multichannel A-Format (or B-Format) audio stream to the corresponding converter.

3.3 Sampling Ambisonic Decoding with *SADN*

An effective method of ambisonic decoding is Sampling Ambisonic Decoding (SAD), where a portion of the encoded sound field is sampled and reproduced by a loudspeaker. It is most effective with *optimal* loudspeaker layouts, where all loudspeakers are equidistance from center, and loudspeakers are evenly spaced [2]. Using (2) we form a matrix using the angular coordinates of loudspeakers:

$$\mathbf{Y}_N = [\mathbf{y}_1(\Theta_1), \mathbf{y}_2(\Theta_2), \dots, \mathbf{y}_N(\Theta_N)]^T \quad (5)$$

To assist the preservation of acoustic energy, an additional coefficient is added, which depends upon whether the loudspeaker array is 2D or 3D. So, we obtain the coefficients

performed by the following calculation [2]:

$$\mathbf{D}_N = \sqrt{\frac{E_D}{L}} \mathbf{Y}_N \quad (6)$$

Where D corresponds to the loudspeaker array's degree of dimensionality so that $E_{3D} = 4\pi$ and $E_{2D} = 2\pi$.

3.4 Dual Band Ambisonic Decoding with DBDN

Researchers such as Heller and Benjamin [20, 12] have developed optimization schemes for Gerzon's localization assessments laid out in his metatheory [4]. By splitting the incoming B-Format signal into two frequency bands, two key localization methods can be optimized. In the low frequencies ($\leq 750\text{Hz}$), interaural time differences (ITD) dominate localization, and in higher frequencies ($750\text{Hz} \leq f \leq 5\text{kHz}$), interaural level differences (ILD) dominate.

Gerzon presented tests for measuring the extent to which these cues are accurately functioning in multi-channel systems [4]. In [21] and [12], these cues are maximized by different weightings of the frequency bands. This method has been implemented with DBDN, which works well with both regular and irregular loudspeaker arrays. While localization accuracy of DBDN is much higher than SADN, it is more computationally costly.

3.5 A Decoder "Boilerplate" with DecodeN

While experimenting with ambisonic processing, users may consider constructing their own decoders. Decoders such as DBDN have a clear signal paths which are designed to emphasize certain aspects of BFormat signals, similar decoders can be recreated in ChucK level code via DecodeN. The main goal of DecodeN is to provide savvy (or not so savvy) users with the tools to experiment.

From a signal processing perspective, DecodeN operates exactly like the aforementioned SADN, with a *re-encoding* method of decoding detailed in Eq.(6). The clear distinction between SADN and DecodeN is that users must provide DecodeN with spherical harmonic coefficients themselves, whereas SADN will calculate SN3D harmonics given spherical coordinates. This method of control allows for custom weighting systems, most beneficial for static site-specific decoders. DecodeN is optimal for users looking to experiment with and push the boundaries of current ambisonic literature.

4. EXAMPLES

All of the following examples utilize Chumbi's components to achieve real-time ambisonic processing.

4.1 HandInSpace, the Ambisonic iOS Interface

Using skeleton tracking provided by the iOS app DataOSC, a simple interface layer was created to control virtual source positioning. This is done by linking the coordinate data of both hands to corresponding sounds, and positioning them accordingly.

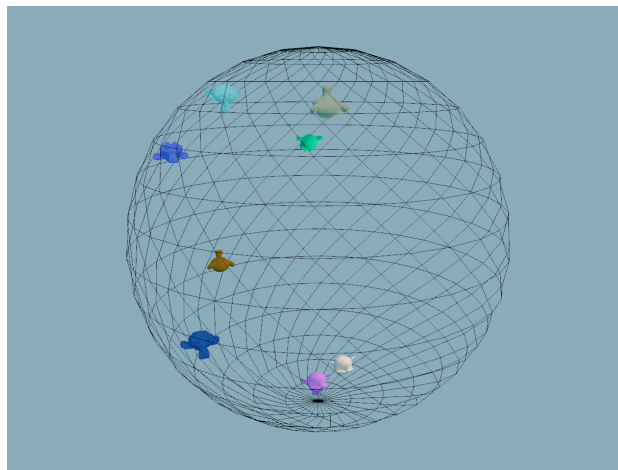


Figure 2. Skeleton tracking used within DataOSC.

Figure 3. Updating of granulator positioning in AmbiGranny

```

fun void updateEncoder(AmbiGranulator g, Encode3 encoder)
{
    while(true)
    {
        // changing one's position
        // is done with a single function call
        encoder.pos(g.azimuth, g.zenith);
        // update positioning every 5ms
        5::ms => now;
    }
}

```

4.2 SpatialSuzanne, a ChuGL Visualizer

Using ChucK's audiovisual framework ChuGL [22], a program to visualize the positioning of sources was easy to accomplish. Users can easily see their source positioning as an aid during live performance or for backing visuals. Gerzon's vector models [4] have also been applied to this program, allowing users to easily visualize the performance of their decoders.

4.3 AmbiGranny, an Ambisonic Granulator

AmbiGranny is an ambisonic granulator which positions grains within space according to a *spread* parameter. The user controls AmbiGranny through a laptop or external keyboard, which they can use to change pitch, length, positioning and spread of grains. An example of Chumbi's usage in AmbiGranny is shown in Fig.3. As mentioned previously in the introduction, large arrays of UGens allowed ambisonic processing, which were used in early versions of AmbiGranny. Using Chumbi, AmbiGranny has reduced it's original 1085 lines of code to 350.

4.4 Usage in Chunity

As of 2018 the Unity game engine is capable of running ChucK [23], allowing users to mix audio and visual programming using Chunity, a package available on the Unity Asset Store. Since Chunity is capable of running Chugins, users can utilize Chumbi within Unity projects. Various implementations have been tested, including a scene where

users can leverage spatial positioning of themselves while performing together.

Acknowledgments

Sincere thanks is extended to everyone who supported this work: to () and () for their help when formatting Chumbi as a ChuMP package, and to () and () for their encouragement and guidance.

5. REFERENCES

- [1] J. Daniel, R. Nicol, and S. Moreau, "Further Investigations of High Order Ambisonics and Wavefield Synthesis For Holophonic Sound Imaging," in *Proc. of the 114th Convention 2003 March 22–25 Amsterdam, The Netherlands*, 01 2003.
- [2] M. Frank. and F. Zotter, *Ambisonics: A Practical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality*. Creative Media Partners, LLC, 2020. [Online]. Available: <https://books.google.com/books?id=LET2zQEACAAJ>
- [3] V. Pulkki, "Virtual Sound Source Positioning Using Vector Base Amplitude Panning," *Journal of The Audio Engineering Society*, vol. 45, pp. 456–466, June 1997.
- [4] M. Gerzon, "General Metatheory of Auditory Localization," *Journal of The Audio Engineering Society*, no. 3306, March 1992.
- [5] D. G. Malham, "Spherical Harmonic Coding of Sound Objects - The Ambisonic 'O' Format," *Journal of The Audio Engineering Society*, no. 1919, June 2001.
- [6] B. Wiggins and M. Dring, "AmbiFreeverb-2 Development of a 3D Ambisonic Reverb With Spatial Warping and Variable Scattering," *Journal of The Audio Engineering Society*, no. 2-3, July 2016.
- [7] C. Oreinos, J. Buchholz, and J. Mejia, "Effect of Higher-Order Ambisonics on Evaluating Beamformer Benefit in Realistic Acoustic Environments," 10 2013, pp. 1–4.
- [8] P. Lecomte, G. Philippe-Aubert, G. A. A. Berry, and C. Langrenne, "Directional Filtering of Ambisonic Sound Scenes," *Journal of The Audio Engineering Society*, no. P7-1, July 2018.
- [9] J. Meyer and G. Elko, "A Qualitative Analysis of Frequency Dependencies in Ambisonics Decoding Related to Spherical Microphone Array Recording," *Journal of The Audio Engineering Society*, no. 6-1, 2016.
- [10] G. Wang, P. R. Cook, and S. Salazar, "ChucK: A Strongly Timed Computer Music Language," *Computer Music Journal*, vol. 39, no. 4, pp. 10–29, 2015. [Online]. Available: <http://www.jstor.org/stable/43829289>
- [11] P. M. A., "A Unified Theory of Horizontal Holographic Sound Systems," *Journal of The Audio Engineering Society*, vol. 48, pp. 1155–1182, December 2000.
- [12] A. J. Heller, "DESIGN AND IMPLEMENTATION OF FILTERS FOR AMBISONIC DECODERS," in *Proceedings of the 1st International Faust Conference (IFC-18), Mainz, Germany, July 17–18, 2018*, 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:51695060>
- [13] J. Daniel and S. Moreau, "Further Study of Sound Field Coding with Higher Order Ambisonics," in *Proc. of the 116th AES Convention 2004 May 8–11 Berlin, Germany*, 05 2004.
- [14] G. Williams, *Fourier Acoustics: Sound Radiation and Nearfield Acoustical Holography*. Academic Press, 1999. [Online]. Available: <https://books.google.com/books?id=TN76mCeHwAUC>
- [15] C. Nachbar, F. Zotter, E. Deleffie, and A. Sontacchi, "AMBIX -A SUGGESTED AMBISONICS FORMAT," *Ambisonics Symposium 2011*, 07 2011.
- [16] M. Gerzon, "The Design of Precisely Coincident Microphone Arrays For Stereo and Surround Sound," *Journal of The Audio Engineering Society*, no. L-20, March 1975.
- [17] middlicott charles and wiggins bruce, "development of ambisonic microphone design tools—part 1," *journal of the audio engineering society*, no. 489, october 2018.
- [18] F. Lopez-Lezcano, "THE SPHEAR PROJECT UPDATE: REFINING THE OCTASPHEAR, A 2ND ORDER AMBISONICS MICROPHONE," 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:209478757>
- [19] —, "The *SpHEAR Project Update: The TinySpHEAR and Octathingy Soundfield Microphones," 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:197859719>
- [20] A. Heller, E. Benjamin, R. Lee, and P. Litoral, "A Toolkit for the Design of Ambisonic Decoders," in *Proceedings of the Linux Audio Conference of 2012*, 2012. [Online]. Available: <https://api.semanticscholar.org/CorpusID:7126173>
- [21] D. Jérôme, R. Jean-Bernard, and P. Jean-Dominique, "Ambisonics Encoding of Other Audio Formats For Multiple Listening Conditions," *Journal of The Audio Engineering Society*, no. 4795, September 1998.
- [22] A. Zhu and G. Wang, "ChuGL: Unified Audiovisual Programming in ChucK," pp. 351–358, September 2024. [Online]. Available: http://nime.org/proceedings/2024/nime2024_52.pdf
- [23] J. Atherton and G. Wang, "Chunity: Integrated Audiovisual Programming in Unity," in *Proceedings of the International Conference on New Interfaces for Musical Expression*. Blacksburg, Virginia, USA: Virginia Tech, June 2018, pp. 102–107. [Online]. Available: http://www.nime.org/proceedings/2018/nime2018_paper0024.pdf