# Volume Rendering for Pixar's *Elemental*

Julian Fong
jfong@pixar.com
Pixar Animation Studios

Two scenes from *Elemental* showing a wide variety of volumetric characters. ©Disney/Pixar.

## ABSTRACT

This work presents recent updates to volume rendering in RenderMan, the production renderer used at Pixar. With the advent of increasingly complicated volumetric assets such as those seen in Pixar's films *Soul* and *Elemental*, our aggregate volume renderer required changes to handle complex water based characters; optimizations for complicated scenes; better handling of transform blur; and improved support for volumetric matte holdouts. These changes allowed us to render an immense number of complicated and detailed volumetric characters in *Elemental*.

**ACM Reference Format:**
Julian Fong. 2023. Volume Rendering for Pixar's *Elemental*. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '23 Talks), August 06–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3587421.3595433

## 1 INTRODUCTION

The current generation of RenderMan, first used in Pixar's *Finding Dory* [Christensen et al. 2018], has deployed two full implementations of volume rendering. The first implementation was designed for the requirements of *Finding Dory*: many nested dielectric surfaces with subsurface and single scattering. While this system was fully general and capable of rendering any volumetric setup, it was not optimal for more common production cases: large numbers of overlapping volumes (clouds) not bound to the interior of dielectric objects. A second implementation was designed to tackle these cases, which we call *aggregate volumes.* Both of these systems are described in more detail in the 2017 volume rendering course notes [Fong et al. 2017]. Over time, aggregate volumes has become the primary volume rendering system used at Pixar, but with the advent of increasingly complicated volumetric sets and characters, work was needed to extend this system.

The data structure behind aggregate volumes is an octree encompassing all volumes. Each node stores a list of volumes that partially or fully overlap the region, as well as relevant precomputed metadata (typically the extinction extrema) gathered over such volumes. Nodes are recursively subdivided as long as a given heuristic is true. Any ray that integrates the volumes in the scene performs an optimal line walk of the nodes until a scattering event is determined; determination of this event freely makes use of the available metadata from traversed nodes. Key to the efficiency is that the boundary behavior of the volume containers is not relevant to the render: rays will not bend. This allows construction of an acceleration structure that facilitates traversal algorithms using the node metadata to entirely skip over volumes without needing to interrogate their properties, minimizing the need to expensively query individual volumes during traversal.

## 2 CONTRIBUTIONS

For the production of Pixar's films *Onward*, *Soul*, *Luca*, *Lightyear*, and culminating with *Elemental*, we developed extensions to aggregate volumes that met the requirements of production while maintaining the efficiency of the system.

*Binding volumes to dielectric interiors:* the initial design of aggregate volumes did not permit heterogeneous participating media constrained to the interior of a dielectric object precisely because the boundary behavior of such an object could not be taken into account. Thus we developed a hybrid approach that adopted ideas from the older volume rendering implementation. First, we allow the construction of any number of aggregates, going beyond the usual single global aggregate. Each such aggregate is named with a unique string. Second, materials bound to dielectric materials may specify an *interior aggregate* by name. When a ray transmits

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*SIGGRAPH '23 Talks, August 06–10, 2023, Los Angeles, CA, USA*
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0143-6/23/08.
https://doi.org/10.1145/3587421.3595433

through such a material, the integrator switches from traversing the default global aggregate to only the named interior aggregate bound to the surface. With the aid of a small stack maintained per ray, we revert to the global aggregate when the ray exits the same surface. Key to this scheme is separation of the specifications of the volume geometry and its material properties from the surface geometry and material: both sets are specified independently from each other, which was not possible prior to this scheme. This system was first deployed successfully to render a translucent monster in *Onward*, used extensively for the oceans in *Luca*, and was used to create all the water characters in *Elemental*.

*Octree optimization:* The astral plane in *Soul* proved to be a challenge for the scalability of our aggregate volumes; in particular, the Hall of Everything contained a vast number of volumetric objects, requiring optimization. Most of this was focused on tweaking the octree subdivision heuristic by adjusting the arbitrary cutoff value $T$ in $(\max(R) - \min(R)) \cdot \texttt{diag}(R) > T$ to be $1/log(0.5) = 1.442$. Our rationale for adjusting this value is to better approach the theoretical ideal of one density evaluation per aggregate segment. This greatly increased the number of nodes, which in turn required a restructuring of the data structures in order to improve cache occupancy. By changing the octree to being implicit, eliminating pointers, and disallowing threads from traversing and updating the octree simultaneously, we were able to reduce our memory consumption by more than 2x, which was more than enough to offset the widened branch factor. This work was sufficient to speed up complex shots by more than 30 percent in overall runtime.

*Temporal extensions:* the initial implementation of aggregate volumes did not handle transformation motion blur. We initially avoided this issue by converting to deformation motion blur; however, the requirement of baking velocity data for every asset became untenable. We solved this problem by extending our octree nodes with temporal occupancy data. Extending the octree to a four-dimensional analog (where every node has 16 children) would have increased the memory constraints unacceptably. We chose a compromise: each node was extended with two values, `minTime` and `maxTime`, the minimum and maximum time when the summed maximum extinction was non-zero. This required extending the octree build stage to compute metadata over the full range of motion of all overlapping volumes. During ray traversal, the time associated with the ray was compared against the time extents to quickly accept or reject a scattering event. It also proved critical to adjust the octree node split criterion by adding a compensating factor of time occupancy: $(maxTime - minTime) \cdot (\max(R) - \min(R)) \cdot \texttt{diag}(R) > T$. Further work was also required to better handle cases where fast motion blur of camera cancelled out fast moving volumes, including changing the native transformation space of the octree.

*Visibility extensions:* RenderMan supports a flexible system which not only includes visibility of geometry to three categories of rays (camera, shadow, and indirect), but also allows any geometry subset to be specified as visible to any light subset, separately for direct lighting and shadows. We cannot hope to build metadata for all such possible scene subsets. As a compromise, we extended nodes to include three sets of metadata: one each for camera, shadow, and indirect visibility. This provided enough flexibility for lighting on *Elemental*. More complicated visibility queries were handled by querying individual volumes during scattering events, which was suboptimal but not common. With these changes, the core metadata stored in our octree node became three pairs of floats (per-visibility extinction ranges) and two bytes (time interval), along with the list of volumes. This was an acceptable balance between artistic flexibility and memory required for the aggregate volume octree.

*Alpha channel handling:* generating a clean alpha channel from volumes is important for any compositing work. Our volume light sampling techniques run the gamut from density sampling to product importance sampling using virtual density segments [Wrenninge and Villemin 2020]. Unfortunately, any such technique whose importance is not proportional to density produces suboptimal alpha results. Our approach to generating cleaner alpha channels is to integrate our camera rays twice. The first pass performs all lighting calculations as usual, but with alpha channel output disabled. In the second pass, camera rays up to the first hit are integrated without any light calculations, with alpha channel enabled. Only delta tracking (or other sampling technique with importance proportional to volume transmittance) is allowed on this second integration.

*Matte holdout workflow:* the possibility of volumes holding out other volumes increases the complexity of alpha channel output. Initial implementations of this workflow were only correct when the matte and non-matte volumes did not overlap. In order to correctly resolve these setups, we augment the second alpha-only volume integration pass to simultaneously compute an estimate for the *matte correction factor* $\alpha_m = \sum(1-\tau_m) \cdot T_{n-1}$ alongside the estimate for the transmittance $T_n = \prod_{i=1}^{n}(\tau)$; $\alpha_m$ can then be subtracted from the final alpha to correctly account for held out volumes.

## 3 FUTURE WORK

Volumetric light sources [Villemin and Hery 2013] were initially used on *Elemental*. Unfortunately, we were not able to converge renders quickly enough because of stylistic choices made for the fire characters — their volumes had very high extinction coefficients. Importance sampling for next event estimation of points within the flames did not take into account the visibility term, which often meant that sampled light positions ended up being shadowed by the rest of the character. Future improvements to light sampling to deal with such cases will need to incorporate a precomputed visibility term.

Our aggregate volume implementation has recently been implemented on GPU hardware for the next-generation RenderMan XPU renderer. We have found that it works well there, with certain design tweaks necessitated by the very large ray batch size that is needed in order to take full advantage of GPU compute units.

## REFERENCES

Per Christensen, Julian Fong, Jonathan Shade, et al. 2018. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering. *ACM Transactions on Graphics* 37 (08 2018), 1–21. https://doi.org/10.1145/3182162

Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. 2017. Production Volume Rendering: Siggraph 2017 Course. In *ACM SIGGRAPH 2017 Courses*. 1–79.

Ryusuke Villemin and Christophe Hery. 2013. Practical Illumination from Flames. *Journal of Computer Graphics Techniques (JCGT)* 2, 2 (31 December 2013), 142–155.

Magnus Wrenninge and Ryusuke Villemin. 2020. *Product Importance Sampling of the Volume Rendering Equation using Virtual Density Segments*. Technical Note 20-01. Pixar Animation Studios. https://graphics.pixar.com/library/CandidateSampling/paper.pdf