Rasterizing Volumes and Surfaces for Crowds on *Soul*

Sasha Ouellet
Pixar Animation Studios

Daniel Garcia
Pixar Animation Studios

Stephen Gustafson
Pixar Animation Studios

Matt Kuruc
Pixar Animation Studios

Michael Lorenzen
Pixar Animation Studios

George Nguyen
Pixar Animation Studios

Grace Gilbert
Pixar Animation Studios

Figure 1: Shots with hundreds of volumetric crowd characters required an efficient volume generation pipeline. ©Disney/Pixar.

ABSTRACT

In order to produce shots with hundreds of multi-volume crowd characters for *Soul*, we could not rely on the same I/O heavy pose-cache pipeline used for the hero characters [Coleman et al. 2020]. Our rendering and systems teams rated the total necessary storage for the "soul world" at over 100 TBs for an average of two hero characters per shot.[1] For expansive crowds of these characters to hit the same volumetric look while avoiding this I/O limitation, two new render-time technologies were developed. The first leveraged an existing volume rasterizer to pose volumes at render-time, informed by a lattice deformation. The second allowed for rasterization of surface primvars to be delivered to the volume shaders.

ACM Reference Format:
Sasha Ouellet, Daniel Garcia, Stephen Gustafson, Matt Kuruc, Michael Lorenzen, George Nguyen, and Grace Gilbert. 2020. Rasterizing Volumes and Surfaces for Crowds on *Soul*. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '20 Talks), August 17, 2020.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3388767.3407374

[1]This projection exceeded the actual peak usage of 80 TBs.

SIGGRAPH '20 Talks, August 17, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7971-7/20/08.
https://doi.org/10.1145/3388767.3407374

1 VOLUME DEFORMATION

In addition to the I/O limitations of pursuing a pose-caching pipeline, the Houdini hero volume generation time would have been a large renderfarm cost if applied to crowds. With our internal volume rasterization library, REVES[2] [Wrenninge 2016], we built a pipeline around rest pose volumes and deformed lattices. This allowed us to generate implicit volumes at render-time in order to prevent volume data from hitting network storage.

As a part of the crowds character build, a lattice was generated for each character's volumes - body, hair, face details, and accessories[3]. The lattice is deformed by the UsdSkel crowds pipeline [Yen et al. 2018], which yields the lattice point motion samples to a RenderMan implicit field plugin. With the deformed lattice and a rest volume from disk, the plugin performs the voxelization of each of the fields at render-time including a velocity field computed from the velocity vectors of the points. The output velocity field is later sampled to support deformation motion blur of the volumes. Additionally,

[2]REVES is a volumetric implementation of the REYES algorithm.
[3]The resolution of each lattice is 15x15x15, which strikes a balance between preservation of detail from rasterization and performance.

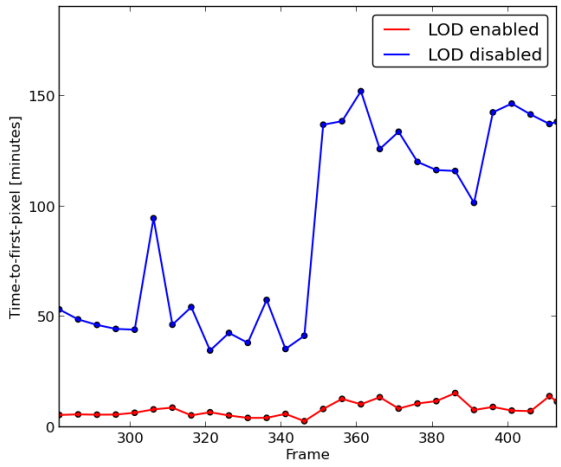

Figure 2: Time-to-first-pixel comparisons of voxel resolution LOD for a particular production shot. ©Disney/Pixar.

minmax fields are generated for optimal tracing post-rasterization via RenderMan's volume tracking algorithms [Fong et al. 2017].

Rasterizing the input rest pose at full voxel resolution was a high cost for pre-pixel computation time. This slowed down iteration time for lighters, requiring use of our stochastic LOD technology for tuning the output field resolution [Cook et al. 2007].

2 GEOMETRY RASTERIZER

While these crowd characters were mostly built from volumes, the eyes were meshes. The volumetric faces do not provide the same occlusion as a surface, resulting in a "floating" eyeball look. Computing the occlusion culling could be easily done with a Z-buffer — however because our characters were not opaque and overlapped in screen space, it was necessary to produce a unique buffer per character. This requirement barred the use of RenderMan to generate the signals, as it assumes one render context per process, not hundreds. We developed a CPU rasterizer[4] to produce in-memory textures. With this embeddable renderer, Z-depth is rasterized to perform a depth test against the eye surface for presence masking.

Additionally, certain fields of the rest pose volumes used for shading could not be easily rasterized by REVES. Camera independent fields (i.e. density or masking fields) were trivial, but some signals (like the gradient fields) were frustum aligned. Additionally, the REVES rasterization cost of these fields contributed to growing time-to-first-pixel concerns. We used the geometry rasterizer to render the necessary analogous surface properties of each volume's representative mesh to textures, which could then be wired into the volume shaders as if they were coming from field data. This enabled our shading artists to construct a shading network that could easily be used for both the hero and crowd characters, regardless of where the signals were coming from. Additionally, a separate surface shader for emulating the volumetric look leveraged an alpha texture produced by the rasterizer.

[4]We chose not to develop a GPU rasterizer since our farm is primarily CPU-based.

It was advantageous to perform the rasterization just-in-time in order to avoid synchronization issues with the crowds pipeline. Furthermore, these signals were required for light shaping, meaning rasterization needed to be a preprocess of the render as opposed to offloading the responsibility to comp. This geometry rasterizer has a limited feature set and is not meant to replace the capabilities of a REYES renderer; it is simply a low cost one-sample rasterizer designed to provide input to other parts of the scene.

2.1 In-Memory Textures

The system needed to support several 512x512 textures per character, per frame.[5] In order to avoid the I/O cost of writing and reading several thousand textures per shot, the rasterized textures were kept in-memory as attributes of the geometry, which could be moved to the RenderMan Rtx texture interface. We wrote an Rtx plugin to fulfill tile fill requests from the buffers, which also supported generating mipmaps for blurred texture access. Texture sampling the edges of the volumes required a dilation of the rasterized signals. We composited higher resolution mips on top of lower resolution mips as a flooding mechanism that was fast and more temporally coherent than OpenImageIO's push-pull algorithm.

3 CONCLUSIONS

Through integration of rasterization techniques into our path traced renderer's scene initialization, we were able to support vast amounts of volumetric crowd characters in *Soul*. REVES provided a means to pose crowds of volumes at render time, while the geometry rasterizer yielded the necessary shader inputs in order to closely match the hero look at a fraction of the cost.

ACKNOWLEDGMENTS

Thanks to Magnus Wrenninge for the development of REVES, Julian Fong and Stephen Friedman for RenderMan support, Michael Rice for Field3D support, Steve LaVietes for Katana support, and Frankie Eder for geometry rasterizer optimizations.

REFERENCES

Patrick Coleman, Laura Murphy, Markus Kranzler, and Max Gilbert. 2020. Making Souls: Methods and a Pipeline for Volumetric Characters. In *ACM SIGGRAPH 2020 Talks (SIGGRAPH '20)*. Association for Computing Machinery, New York, NY, USA, 2.

Robert L. Cook, John Halstead, Maxwell Planck, and David Ryu. 2007. Stochastic Simplification of Aggregate Detail. In *ACM SIGGRAPH 2007 Papers* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 79–es. https://doi.org/10.1145/1275808.1276476

Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. 2017. Production Volume Rendering: SIGGRAPH 2017 Course. In *ACM SIGGRAPH 2017 Courses* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article Article 2, 79 pages. https://doi.org/10.1145/3084873.3084907

Magnus Wrenninge. 2016. Efficient Rendering of Volumetric Motion Blur Using Temporally Unstructured Volumes. *Journal of Computer Graphics Techniques (JCGT)* 5, 1 (31 January 2016), 1–34. http://jcgt.org/published/0005/01/01/

Jane Yen, Stephen Gustafson, Aaron Lo, J. D. Northrup, and Lana Sun. 2018. Other Worldly Crowds in Coco. In *ACM SIGGRAPH 2018 Talks* (Vancouver, British Columbia, Canada) *(SIGGRAPH '18)*. Association for Computing Machinery, New York, NY, USA, Article Article 19, 2 pages. https://doi.org/10.1145/3214745.3214796

[5]This is the average resolution for mid-ground characters. We used the screen-space size of the character to procedurally determine the necessary resolution, ranging from 2048x2048 to 256x256.