# Organizing the Fuzziest Concerts for Trolls World Tour

Kuan-Ting Lu
DreamWorks Animation
tlu@dreamworks.com

**Figure 1: A hard rock concert in *Trolls World Tour.* ©2020 DreamWorks Animation. All rights reserved.**

## ABSTRACT

The unique, stylized look of *Trolls World Tour* presented complexity challenges to our rendering pipeline. Each asset was covered in high-density fuzz, with millions of curves processed each scene. We revisited the way we handle curve geometries with new optimization methods and a new caching system to achieve interactive loading speeds and scalable render capacity in our lighting tools.

## CCS CONCEPTS

• **Computing methodologies** → *Computer graphics.*

## KEYWORDS

lighting, instancing, memory, caching

**ACM Reference Format:**
Kuan-Ting Lu. 2020. Organizing the Fuzziest Concerts for Trolls World Tour. 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/10.1145/3388767.3407365

## 1 INTRODUCTION

The world of Trolls required high scene complexity, with each asset covered in high-density fuzz, from woolly plants growing on carpet forest floors to the wild growing hair of each individual Troll. In the final hard rock sequences of the film, millions of high-fidelity curves rested on the fiery concert venue and on a crowd of ten thousand Trolls, totaling hundreds of millions of CVs (Figure 1). On the first *Trolls* movie, geometry shaders were used to achieve the fuzz. In order to achieve the ideal look, we needed the ability to have art-directable strands of hair that could be viewed at multiple stages in the pipeline. It was quickly decided that each hair needed to be distinctive curve geometry.

Our first approach was to leverage the scene processing power of our lighting toolset to create a procedural solution to geometry-based fuzz directly in the lighting tool. Artists defined several clumps of curves that acted as the template of the look of an asset's fuzz. The clumps of curves were instanced at multiple levels and distributed on the surface of the asset's geometry. The approach was deferred to render time, which saved memory and disk space, and kept artist's working sessions lightweight. However, the nested instancing distribution was limited and it was difficult to art direct and achieve the desired look. The curves needed to be authored traditionally by artists.

As production progressed to finalized assets and fully populated scenes, the complexity of the scenes grew significantly, as each asset was accompanied by dense, high-resolution fuzz. Trolls were characterized by a head full of hair and felt-like skin composed of millions of curves (Figure 2). Initial renders ran out of memory and it was apparent that further optimizations and a scalable approach to moving data through the pipeline was needed. The lighting department needed a way to view and work interactively with these heavy scenes, and see the fuzz as it was intended by upstream departments.

**Figure 2: Trolls' felt-like skin covered with fuzz.**

## 2 SCENE DESCRIPTION

To manage the vast amount of curve data, we leveraged our internal Procedural USD schema [Blevins and Murray 2018] that could point to any arbitrary data and be passed along to the renderer or read into the lighting tool. This allowed dynamic choices between speed and memory by choosing to pay the cost of evaluating the curves interactively or during render time. Since the underlying data was not part of the USD stage, the USD files were compact and stage loading times fast. As USD procedural data was loaded into the lighting tool, custom operators resolved them into raw curves data for viewing and editing. We then created a custom procedural for our renderer that was able to reference this data directly during rendering, which prevented copying of the data, improving speed and reducing peak memory usage.

## 3 OPTIMIZING CURVES

The vast number of curves burdened interactivity for artists and were well beyond the memory limitations of our rendering capacity. Traditional optimization methods such as frustum culling were insufficient, as in many cases there were millions of curves included within the frustum. Given that Trolls are tiny creatures, large objects such as the ground plane or a massive tree could include a significant amount of curves that are not visible. Other traditional workflows such as instancing and LOD partitioned crowds were used where possible, however given the vast diversity and complexity of the Trolls world it was a limited solution. Solely relying on partitioning would be extremely cumbersome.

Expanding the traditional workflow, we implemented a series of curve operators that ran on every fur and hair procedural by default. We extended frustum and volume culling to a much more precise level within each geometry, culling out individual strands of hair on each asset. We added an occlusion based culling that utilized the same raytracing method as our renderer, and culled out invisible curves. These provided mass culling solutions at a more granular level and reduced wasted resources, especially for massive objects that extend well outside the frustum. For LOD, in addition to density reduction operations, we added culling methods granular to the CV level, able to simplify individual curves. Artists have granular control over the density of fuzz and complexity of each individual curve to obtain the desired look. Each of these operators ran deferred for higher interactivity and the ability to pick and choose any combination of the operators available. We exported these same operators to USD via a custom USD schema named OpAPI. This allowed artists across all departments to render the optimized scenes outside of the lighting tool in our USD rendering engine, and maintains parity between what the lighting department and upstream departments (layout, animation, crowds) see in their own renders.

## 4 FAST DYNAMIC CACHING

Resolved curve geometry could additionally be cached to disk. Our default lighting package ships with a caching mechanism that caches attribute data to binary data in memory. We found the serialization of binary data to be a bottleneck in cache performance, and caching to memory limited cache usage to each work session. We extended the caching mechanism with our own disk based, self evicting cache. While the shipped cache took more than 77 seconds to serialize around 2 gigabytes of data, our cache reduced that same serialization to about 1 second.

As our multi-threaded scene traversal processed each curve geometry, the optimizations were applied, results cached, and the memory released while new geometry began the process again. This method was able to reduce the peak memory usage considerably and also reduced disk usage while retaining parallelism. Once available, the cache accelerated loading of geometry and subsequent renders became much faster and used significantly less memory; it even allowed millions of curves to be loaded and viewed at interactive speeds. Lighters could view the curves in context interactively, and further optimization or culling decisions could be made. The system allowed the same asset's cache to be shared between each session, and potentially across shots and sequences. As the render farm could access the same cache directory, the same benefits applied to farm renders. Static assets were further optimized and only generated once between all frames. With the new cache in place, a relatively small environment consisting of about 400 million CVs saw scene build times reduced from 47 seconds to 21 seconds, and peak memory usage reduced from 41 to 15 gigabytes.

## 5 CONCLUSION

The complexity of *Trolls World Tour* challenged us to think about memory from the beginning of the pipeline. We needed a scalable pipeline to handle the way data was authored and passed, previewed across departments, loaded into the lighting tool, and rendered. We developed multiple points of optimizations, instancing, culling operators, disk caches, and custom procedurals which all needed to work together. As complexity in future films increase, these tools could provide a foundation for future optimized workflows.

## ACKNOWLEDGMENTS

Alex Gerveshi, Sean Looper, Ali Khouzadi, Niel Lopez, Josh Miller, Cherry Chen, Alan Blevins, Garrett Broussard, Jonathan Ciscon, Chris Edwards, Chris Sprunger, Christy Page

## REFERENCES

Alan Blevins and Mike Murray. 2018. Zero to USD in 80 Days. In *ACM SIGGRAPH 2018 Talks* (Vancouver, Canada) *(SIGGRAPH '18)*.