Engineering Full-Fidelity Hair for Incredibles 2

Andrew Butts
andru@pixar.com
Pixar Animation Studios

Ben Porter
bporter@pixar.com
Pixar Animation Studios

Dirk Van Gelder
gelder@pixar.com
Pixar Animation Studios

Mark Hessler
mhessler@pixar.com
Pixar Animation Studios

Venkateswaran Krishna
venkat@pixar.com
Pixar Animation Studios

Gary Monheit
garym@pixar.com
Pixar Animation Studios

Figure 1: A typical display of interactive full-fidelity hair in Presto. ©Disney / Pixar.

ABSTRACT

For *Incredibles 2*, we achieved interactive full-fidelity procedurally-generated hair, yielding artist freedom and productivity surpassing our previous hair tools. We implemented highly parallel algorithms for hair growth and deformation and used fast, modern techniques for graph evaluation, subdivision surfaces, Poisson disk sampling, and scattered data interpolation. Working with full-fidelity hair presented a data scale challenge to our shot pipeline, but we overcame it by allowing a trade-off of control flexibility for speed, maintaining geometric complexity.

CCS CONCEPTS

•**Computing methodologies → Shape modeling;**

KEYWORDS

curve deformation, Poisson disk sampling, parallel computation, GPU

ACM Reference format:
Andrew Butts, Ben Porter, Dirk Van Gelder, Mark Hessler, Venkateswaran Krishna, and Gary Monheit. 2018. Engineering Full-Fidelity Hair for Incredibles 2. In *Proceedings of SIGGRAPH '18 Talks, Vancouver, BC, Canada, August 12-16, 2018,* 2 pages.

DOI: 10.1145/3214745.3214798

1 PLACING HAIR ROOTS

The hair modeling process starts with a Grow operator, which seeds a scalp mesh with hair root locations from which to grow dense hair, subject to scalp density textures. Previously, we were limited to picking root locations uniformly randomly on the scalp subdivision surface's control hull. Artists came up with workarounds involving carefully selecting random seeds to improve root distribution, and carefully choosing the scalp mesh topology to hide artifacts that stemmed from the distortion of the control mesh during subdivision for final renders.

Now, we use OpenSubdiv to refine the scalp mesh and compute a distribution of root points that has the desired density no matter what the shape of the mesh, so artists no longer see hair density artifacts due to scalp mesh topology. We compute scalp samples in parallel.

Because uniform random sampling produces "gritty" looking hair distributions, we implemented a new parallel adaptive Poisson disk sampling technique inspired by [Bowers et al. 2010] using OpenVDB grids for fast spatial neighbor lookups. We filter an initially oversampled distribution in parallel to produce the exact user-prescribed density on the subdivision limit surface, despite being a "dart throwing" approach, due to density compensation multipliers we tabulated experimentally. Other dart throwing approaches [Yuksel 2015] effectively impose a narrow density range for efficiency, but ours is fast even with requested densities ranging from very dense to zero density on the same scalp, as we group regions of similar density into strata, each with a spatial neighbor lookup grid tailored for its density.

2 STATIC SHAPE INTERPOLATION

Once hair root locations are generated, each hair's shape must be interpolated from nearby sparse guide curves. Previously, we required the user to place one guide per scalp vertex in the grow region. Then scalp connectivity determined which guides would combine to calculate one dense hair's shape. This constraint caused interdepartmental overhead, as hair artists often needed to request that the scalp modeler adjust the mesh topology of the scalp to admit more or fewer guides. Also, requiring one guide per vertex placed a high minimum bound on the degrees of freedom a hair modeler had to manage, for even simple short hair styles.

Now, we allow any number of guides to be placed at arbitrary locations on the scalp. This eliminates undesirable departmental coupling but changes the problem into a scattered data interpolation problem. For each hair, our approach grows a hypothetical hair along each nearby guide by gradually transporting a frame from the guide root to the tip, in effect "lofting" the hair root along each nearby guide. These results are blended using one of two approaches depending on the desired look. K-nearest-neighbors distance weighting allows control of smoothness by adjusting K. Natural neighbor interpolation [Sibson 1981] works well when the density of guides over the scalp varies significantly.

3 OPERATORS

Once hairs are grown, they can be deformed by operators specified in any order, or can be used as guides themselves to grow hair. Our deformation operators include Length, Scraggle, Loft, Clump, and Curl. Aside from scalar controls, the inputs for each are limited to an efficient fixed stack of texture and noise inputs, allowing us to avoid implementing a general surface signal evaluation system.

The main performance bottleneck for our hair operators is memory throughput. The full-fidelity representation of the hair can consume hundreds of megabytes, while the operations we perform are arithmetically simple. Our main considerations for performance, aside from parallelism, have been to minimize the number of copies of the hair data, and to minimize the number of passes each algorithm must make over the hair data where possible.

4 FAST NON-DESTRUCTIVE EDITS

Our artists demanded full-fidelity hair editing at interactive rates, as in Weta's Barbershop [Seymour 2015] but would not tolerate the destructive edit workflow of that approach. The robustness of the Presto execution framework allowed us to grow and deform hair using an always-editable graph of operators while still achieving interactive performance. For example, in a chain of operators consisting of a Grow, a Length, a Clump, and Curl, when a user edits a Clump parameter, Presto automatically caches the hair after the Length operator. Updates are faster the closer the edit is to the end of the chain due to this caching, which Presto automatically minimally invalidates when inputs change.

5 MOTION PIPELINE

In shots, animated guides imbue the dense hair with motion using a blended curve deformation approach. Considering only the rest guides and the rest dense hairs, we establish a spatial correspondence from each hair to its nearby guides. Then each hair is hypothetically deformed using each of its nearby guides, and the results are blended according to some procedural weighting. Users can drive the exact choice of weighting to achieve various effects, such as gradually parting hair dynamically as guides separate, or very smooth-looking hair styles.

Where possible, we optimize shot storage requirements by storing only the rest guides, rest dense hairs, and animated guides, performing the motion interpolation at draw time interactively and at final render time. We only incur the cost of writing per-frame animated dense hair to disk when shots contain animated hair operators such as wind.

Guide animation comes from our hair simulator, Taz [Iben et al. 2013], where we consider hair-hair collision and contact only at the guides. Once simulated guides move the bulk of the hair out of collision, any remaining hair-surface collision deformations are stored as sparse data to be overlaid at render time. Guides may also be arbitrarily hand-animated.

6 PERFORMANCE

Previously, our procedurally-generated full-fidelity hair was only viewable as an offline-rendered still image, requiring a lengthy wait and precluding interactive workflows. Now, groomers and shot artists have no such wait time, and freely make edits to hair interpolation parameters, guide animation, and character poses, and see faithful results on dense hair instantly as they make an edit.

7 FUTURE WORK

Optimizing our motion interpolation for the GPU, we expect to achieve even faster speeds for animators. Groomers and shot artists would like to directly manipulate dense hairs at a per-hair-vertex level as they do with guide hairs, as in Weta's Barbershop. This presents a data scale problem we may be able to overcome with a compression scheme such as *ZFP*, which shows negligible quality loss for our hair data at up to 6X compression.

ACKNOWLEDGMENTS

The authors wish to thank Toby Jones and Frank Hanner at Disney Animation Studios for their contributions to our hair motion interpolation approach.

REFERENCES

John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. 2010. Parallel Poisson Disk Sampling with Spectrum Analysis on Surfaces. *ACM Trans. Graph.* 29, 6, Article 166 (Dec. 2010), 10 pages. https://doi.org/10.1145/1882261.1866188

Hayley Iben, Mark Meyer, Lena Petrovic, Olivier Soares, John Anderson, and Andrew Witkin. 2013. Artistic Simulation of Curly Hair. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13).* ACM, New York, NY, USA, 63–71. https://doi.org/10.1145/2485895.2485913

Mike Seymour. 2015. Barbershop at Weta: Sci-tech winner explained. https://www.fxguide.com/featured/barbershop-at-weta-sci-tech-winner-explained/. (2015). Accessed: 2018-05-14.

Robin Sibson. 1981. A brief description of natural neighbour interpolation. *Interpreting Multivariate Data* (1981).

Cem Yuksel. 2015. Sample Elimination for Generating Poisson Disk Sample Sets. *Comput. Graph. Forum* 34, 2 (May 2015), 25–32. https://doi.org/10.1111/cgf.12538