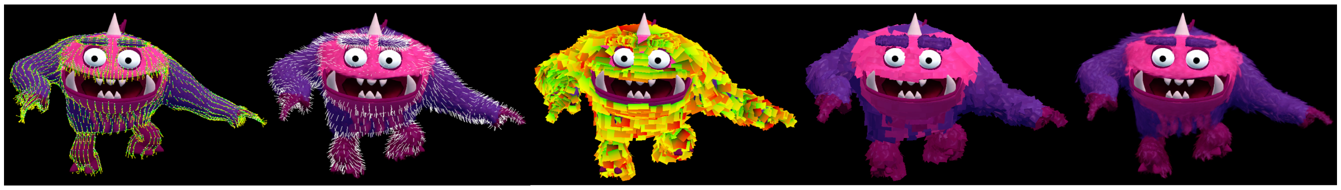
Furry, Floppy, Fuzzy: *Once Upon a Monster's* Fur Pipeline

Peter Demoreuille* Oliver Franzke† Lydia Choy‡
Double Fine Productions



1 Introduction

Sesame Street: Once Upon a Monster required Double Fine Productions to faithfully portray well-known Sesame Street characters and along with a set of furry monster companions. The fur authoring tools, simulation, tessellation, sorting, shading, lighting and rendering techniques created for this goal yielded an efficient and flexible fur system that was used to author and efficiently render fur on every character in the title.

The vast majority of fur rendering in games uses a variation of the "shells and fins" technique, which often leads to a uniform appearance that requires considerable overdraw. Our approach instead renders thousands of individual fur strands, where the simulation and appearance of each strand is customizable. This permits a wide variety of looks, reduces overdraw, provides trivial level-of-detail, and allows the use of higher-quality key-hair simulation to drive strand movement. Fur is rendered using an ad-hoc hair shader, spherical harmonic lighting, and a simplified opacity map for self-shadowing.

1.1 Authoring

Each character's fur is built from several *fur layers*, which generally correspond to local spatial regions (eg, belly, arms, head). Customizability and localized control are paramount, and numerous variables are exposed to drive the appearance of layers and individual strands. Each layer specifies a variety of shader parameters, layout, simulation and sorting controls. Individual per-strand parameters, such as layout density, length, width, curvature, twist and stiffness are encoded in textures, while a vector field exported from Maya is used to determine strand "comb" direction. Curvature influences the amount each strand curves towards or away from the body, twist controls a strand's rotation around its normal, and stiffness controls the degree that each strand reacts to the simulation. Couplied with the size and layout controls, a wide variety of strand layouts can be created, from long matted hair to short twisty fur.

1.2 Simulation

A simulation of a two-segment chain with the root constrained to the character mesh drives fur movement. Constraints provide hair-like movement, particularly segment-to-segment angle-limits and approximate collision checks with the character's skin. The simulation provides gross controls such as damping and constraint strength, while precise control is given by the per-strand stiffness scalar which dampens or exaggerates individual strand movement.

*e-mail:pbd@pod6.org
†e-mail:oliverfranzke@doublefine.com
‡e-mail:lydiachoy@mac.com

Copyright is held by the author / owner(s).
SIGGRAPH 2012, Los Angeles, California, August 5 – 9, 2012.
ISBN 978-1-4503-1435-0/12/0008

1.3 Transparency and Culling

As fur uses alpha blending to create a soft and fuzzy look, strands must be drawn back-to-front. Strand position is determined by barycentric interpolation of mesh vertices and simulation data. An approximation of vertex shader logic calculates a single depth value per strand, and an integer radix sort is used to sort and generate an index array to determine draw order. Backfacing strands generate a sort key that forces them to the end of the index array and allows these strands to be trivially culled, providing an optimization for both the CPU and GPU.

While strand draw order is determined by depth, fur layer sorting is customizable. Layer bounds are initially calculated by their member strands and sorted back-to-front, but additional controls bias and offset the order and are critical when handling edge-cases.

1.4 Geometry, Level-of-Detail and Shading

After strands are sorted and culled, a vertex buffer containing per-strand data is generated. GPU instancing is used to expand strip geometry for each strand, where the number of subdivisions in each strip may be adjusted dynamically. Additional LOD is provided by stochasticly pruning rendered strands: a uniform distribution of strands can be removed by permuting strands offline, and ending updates after the desired number of strands have been processed.

Our shading model supports one shadow casting key light and an unlimited number of fill lights. The key light uses Kajija-Kay[1989] hair shading, and fill lights are projected to (band-2) spherical harmonics. An analytic rim-lighting term rounds out the lighting and adds a pleasant glow to the fur. Shadowing is provided by a simplified opacity map, where fur is rendered back-to-front, accumulating coverage in the alpha-channel. An additional "fake-shadow" term is calculated based on the distance between the fur geometry and character mesh. To compute final fragment color, a base color corresponding to the character skin color is fetched, and luminance, glossiness and transparency scalars are unpacked from a random frame in a per-layer film-strip texture that provides subtle variation.

2 Conclusion

The fur pipeline developed is flexible enough to implement a wide varity of fur styles, utilizes the CPU and GPU efficiently, and is extremely artist-friendly while providing high fidelity results.

References

KAJIYA, J. T., AND KAY, T. L. 1989. Rendering fur with three dimensional textures. In *SIGGRAPH*, ACM, J. J. Thomas, Ed., 271–280.