Build Your Own Procedural Grooming Pipeline

Wanho Choi
Dexter Studios
zelosdev@gmail.com

Nayoung Kim
Julie Jang
Dexter Studios

Sanghun Kim
Dohyun Yang
Dexter Studios

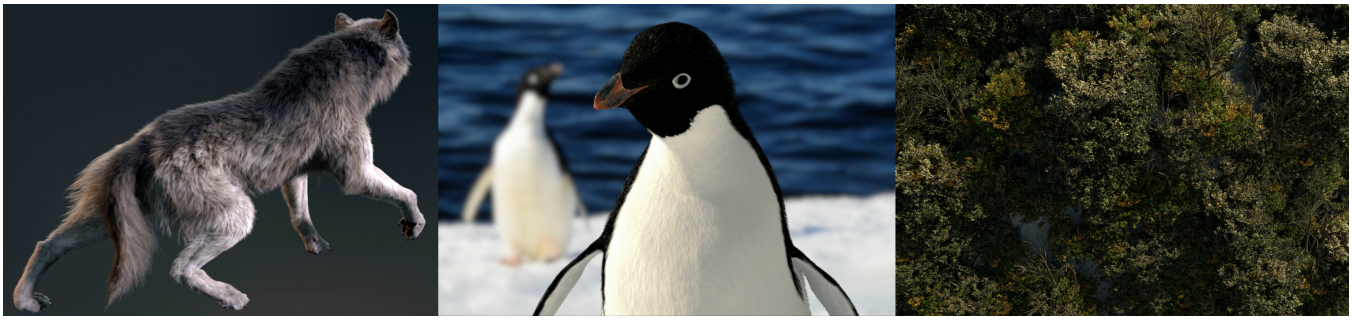

Figure 1: Fur, feathers, and vegetation created through ZENN. ©*Dexter Studios. All Rights Reserved.*

ABSTRACT

Although there is commercially available software for producing digital fur and feathers, creating photorealistic digital creatures under a low budget is still no trivial matter. Because no software could fulfill our purposes at the time of the making of our first movie *Mr. Go*, we decided to develop our own custom solution, ZelosFur [Choi et al. 2013]. While it made possible furry digital creature creation for subsequent projects, this prototypical system lacked the flexibility to easily add new features with backward compatibility and did not provide artists with enough freedom or control over the grooming process. Zelos Node Network (ZENN) is a new procedural solution that allows for quick, easy, and *art-directable* creation of all kinds of body coverings for digital creatures (e.g. fur, feathers, scales, etc.) By extension, it can also be used to create forests, rocks, and verdant landscapes for digital environments. In this talk, we discuss how to design and implement a procedural grooming workflow within ZENN and briefly address our caching and rendering process.

CCS CONCEPTS

• **Computing methodologies** → **Computer graphics**;

KEYWORDS

fur, hair, feathers, environments, procedural geometry

ACM Reference format:
Wanho Choi, Nayoung Kim, Julie Jang, Sanghun Kim, and Dohyun Yang. 2017. Build Your Own Procedural Grooming Pipeline. In *Proceedings of SIGGRAPH '17 Talks, Los Angeles, CA, USA, July 30 - August 03, 2017,* 2 pages.
https://doi.org/10.1145/3084363.3085025

SIGGRAPH '17 Talks, July 30 - August 03, 2017, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5008-2/17/07…$15.00
https://doi.org/10.1145/3084363.3085025

1 PROCEDURAL ASSET GENERATION

ZENN is a Maya plug-in composed of dozens of commands and nodes. As opposed to the one-node-does-all approach of ZelosFur, ZENN was designed to have several modular units in the form of node networks. Artists are able to produce desired output procedurally using the Maya Node Editor. It also allows for the addition of new features with backward compatibility, so it is very accommodating in terms of maintenance.

Every ZENN node has "ZN_" as a prefix. Almost all ZENN nodes have the same input and output data type: inStrands and outStrands. Thus, new nodes can be added easily to the network by insertion between any two sequential nodes. This allows for flexible customization of the final groom.

The typical node network starts with a surface mesh and initial guide curves. These are imported into ZENN, where curves are converted into custom internal data we call "strands." These strands are then bound to the mesh such that they will follow any mesh animation. From the user-provided initial guide curves, an arbitrary number of strands may be generated on the mesh through various sampling and interpolation schemes. The groom is finalized by adding various nodes that affect randomness, clumping, frizz, etc. Curve sets or mesh may then be appended to the strands to create feathers, scales, grass, etc.

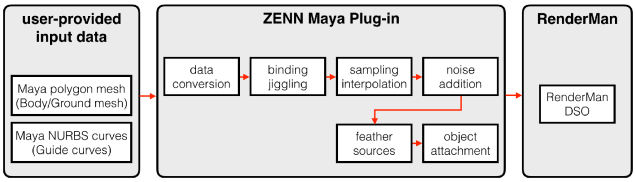

Figure 2: Simplified ZENN workflow diagram.

1.1 Importing Data

Every node network in ZENN starts with a ZN_Import node. The user must provide the ZN_Import node with (1) a Maya polygon mesh to serve as the body/ground surface and (2) Maya NURBS curves to serve as initial guide curves at the rest pose. In general, digital creatures entail a strand count ranging from 100,000 to 1,000,000 on average, but it is not feasible to express such a high number of fur strands in Maya with NURBS curves. Thus, initial guide curves are instead taken into ZENN and converted into internal data we refer to as "strands." We define a strand to consist of a follicle and a curve. The curve represents geometric data, while the follicle contains attributes such as unique ID, texture coordinates, local axes, initial curve length, etc. The output of a ZN_Import node (outStrands) can then be connected as input (inStrands) to any other ZENN node.

1.2 Binding and Simulation

Strands will not move with an animated or otherwise transformed body/ground mesh unless a ZN_Animate node is applied to the network. This node binds strands to their mesh in one of two ways. Users may choose to bind the strands such that they undergo non-deforming, rigid transformations as they follow the animated mesh, or users may wish to apply fake dynamics, making the strands deform with time, but not actually performing any physical simulations. Users can easily control how strands are to "jiggle" through various parameters that take input between 0.0 to 1.0. For the shots in which collision-handling and more robust physical simulations are necessitated, we import the direct results of Maya nHair simulations into the network. Usually, "faking it" with rigid binding is enough because the skin surface deforms dynamically by muscles under it. Accurate simulations tend to only be for hero shots.

1.3 Sampling and Interpolation

A ZN_Generate node generates more strands from input strands through sampling and interpolation. First, it distributes follicles on the surface through one of several provided sampling schemes such as Monte Carlo, Poisson disk sampling, etc. Users are also able to use a density map and/or a remove map to adjust the distribution. After sampling, ZENN calculates the local axes of each follicle using its texture coordinates. To provide stable deformation of strands and feathers while the surface is deforming, it is very important to preserve coherent local axes. While the sampling step is executed only once at the rest frame, interpolation must be computed for every frame. The shape of a strand is determined by a weighted average of its neighboring guide strands. When interpolating, we exclude neighbors whose normals face in the opposite direction of that of the strand in question. Since a ZN_Generate node takes in guide strands as input, and through sampling and interpolation, populates many more strands as output, this enables artists to work in layers through a variable number of ZN_Generate nodes.

1.4 Adding Randomness

After generating a large number of strands from a small number of given strands, we apply a series of modifiers in a ZN_Deform node to reduce artifacts introduced during interpolation. ZENN provides dozens of modifiers such as offset, clumping, bending, rotation, kink, etc. Addition and removal of modifiers as well as the order in which these modifiers are applied may be easily changed and allow for realtime user feedback. Each modifier is controlled by an expression with user-defined paint maps and simplex noise. The unique ID stored at its follicle is used for generating random numbers. As all computations in modifiers are done on the CPU and grooming computations can be somewhat heavy, ZENN enables users to locally update only selected regions for faster artist feedback time.

1.5 Feathers

With the ZN_FeatherInstance node, we can make feathers from fur strands. Users have the option of creating feathers out of curve sets, polygon meshes, or both. These feathers are then instanced and attached to the strands through one of several binding schemes. Feathers can be bound at each follicle to align with the world y-axis, rigidly to the follicle's local axes, along the direction of the strand root-to-tip vector, or fitted to the strand, deforming as the strand deforms. In actuality, all we are doing is instancing "things" onto strands, so if the user inputs a feather, the output will be feathers, but input leaves will become a tree, input trees become a forest, and input grass a meadow, etc. As such, all kinds of body coverings as well as digital environments can be created through ZENN.

2 CACHING AND RENDERING

Published grooming data for the surface mesh at rest pose is combined with an Alembic file containing mesh animations to create fur/feather cache files in our cache generation script. The cache data is sent to our RiProcedural plugin, which transfers follicle attributes such as body mesh uv to the shader. Feathers may each have their own texture map, but users may also wish to define the color of each feather based on the surface mesh texture at its uv coordinate. ZENN provides the option of using either feather texture or body texture as well a blend of both. For feathers, we use low resolution proxy geometry for OpenGL drawing in the viewport and replace it with high resolution geometry at render time.

3 CONCLUSION

In summary, we developed our own custom procedural grooming pipeline in an attempt to tackle fur under a low budget production. ZENN allows for quick, easy, and art-directable creation of all kinds of body coverings for digital creatures, including fur, feathers, and scales. By extension, it can also be used to create forests, rocks, and verdant landscapes for digital environments. Dexter Studios has since used ZENN to create almost all of its digital creatures in some thousand shots, spanning many projects. ZENN continues to undergo regular upgrades and maintenance to this day.

ACKNOWLEDGEMENTS

We would like to thank to all the talented artists, engineers, supervisors, and producers at Dexter Studios for their help and enthusiasm.

REFERENCES

Wanho Choi, Taekyung Yoo, Sanghun Kim, Hyeong-Seok Ko, and Tae-Yong Kim. 2013. Building Efficient Fur Pipeline for a Low Cost Production of Creature-based Feature Film. In *Digital Production Symposium*, Vol. 1.