Art-directing Disney's *Tangled* Procedural Trees

Arthur Shek Dylan Lacewell Andrew Selle Daniel Teece Tom Thompson
Walt Disney Animation Studios

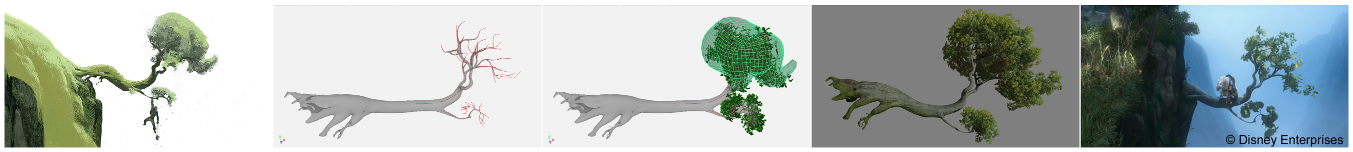

Figure 1: *Left to right: visual development art-direction target (courtesy Kevin Nelson), curve representation, procedural foliage to canopy shell, look dev element render (courtesy Charles Colladay), stylized render from Tangled (sparse leaf variant to support story)*

1 Introduction

Creating stylized trees with hundreds of thousands of leaves is typically a painstaking task that requires hours of artist time. In Walt Disney's animated feature film *Tangled*, we faced the challenge of populating dense forests with animated trees on which artists could quickly iterate to meet an art-directed look. We designed a system of authoring trees based around a language of hierarchical curves. Our system lets artists interactively sketch out a base skeleton representation of a tree and grow procedural twigs and leaves out to a canopy shell by tweaking a limited number of parameters.

2 Curves and Geometry

We determined that the base representation of trunks, branches and twigs would be a system of curves with a notion of hierarchy. This allowed us to write modular pieces of a tree pipeline based around a common structure. Curves also gave us the additional ability to quickly setup animation rigs or dynamic simulations where needed.

To create the desired curve and geometric structure of trees, we designed a number of Maya tools. Certain hero trees called for hand-modeled 3D geometry. We authored a tool called *Skeletor* to extract curves from the model via a mesh skeletonization algorithm [Au et al. 2008]. This allows us to map complex geometry to a network of curves. We also needed software that would allow artists to quickly generate curves to build up a tree. *TreeSketch* is a plug-in that enables artists to interactively draw a hierarchy of curves (see Fig. 1, 2nd image). From hand-sketched or procedurally generated curves, we needed to perform the inverse operation of *Skeletor*, to automatically build geometry from a base curve skeleton. Disney's pipeline necessitated a quad mesh input and we also desired closed, manifold and smooth meshes. Our *Treeman* tool was implemented to meet these specifications, while meeting the challenging requirement of robustly handling many close and competing junctions.

3 Procedural Growth

A standard technique to create procedural branching uses L-systems. We found L-systems require a language and grammar that is unintuitive for artists to describe a desired look. The alternative of hardcoding certain rules and giving artists sliders to dial in multipliers and random seeds also limits the achievable range of looks.

To get around these limitations, we wrote an interactive particle marching engine called *Dendro*. The engine instances particles on the base tree curves according to user exposed parameters. At each timestep, particles can either branch or march out a certain distance in a particular direction, influenced by user parameters. The history of the timestepped particles describe points on a curved path that result in a procedural branching twig structure. Because each point on a twig has an associated lifetime, we can use that information to derive twig width and to instance leaves at desired points.

Dendro's particle method resulted in a number of additional benefits. Artists could further art-direct growth on trees by using canopy shells. These polygonal shells were transformed into voxel regions and used to kill marching particles within a threshold distance, letting artists clip tree growth loosely or tightly to a desired shape. The particle technique also allowed us to include natural tropism effects, such as dealing with environmental factors like gravity, growing towards light, or around obstacles. Artists may dial in wind direction, frequency and amplitude to add procedural animation. Rounding out the core featureset of *Dendro* is the ability for users to control leaf orientation. All parameters are generic enough that they can be used to generate variant trees of the same species simply by changing the random seed. *Dendro* features are exposed to artists at the asset creation level via an interactive frontend within Maya.

4 Rendering

Twigs and leaves were rendered with a Renderman procedural that called the *Dendro* engine. For direct lighting, we used two-sided shaders and deep shadow maps. For indirect lighting and occlusion, we baked static precomputed radiance transfer (PRT) textures with a custom raytracer. The art direction and the forgiving nature of foliage let us optimize the baking aggressively; we used order 3 PRT, simplified leaves to single quads, computed one sample per leaf, and reused the static data even during animation. We used stochastic pruning on the leaves and twigs to cull out appropriate level of detail for rendering large groves of trees. On top of that, we achieved significant acceleration of renders by using a brickmap representation to optimize heavily populated scenes.

Together, this suite of tools has dramatically increased our studio's ability to generate and iterate on largely procedural, yet highly art-directable trees and foliage.

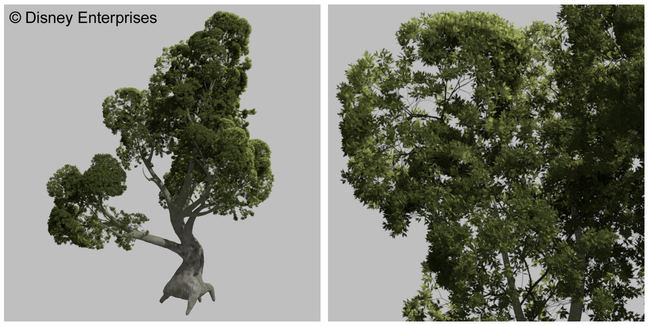

Figure 2: *Left: Tree element from Tangled, 1.02 million leaves (courtesy Larry Wu), Right: Leaf detail*

References

AU, O. K.-C., TAI, C.-L., CHU, H.-K., COHEN-OR, D., AND LEE, T.-Y. 2008. Skeleton extraction by mesh contraction. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, ACM, New York, NY, USA, 1–10.

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007