GPU Graftals: A Method for Stylized Rendering of Fur and Grass

Michael Lester, Rose-Hulman Institute of Technology

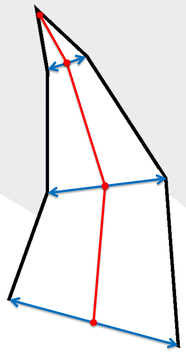
See more at www.realtimemike.wordpress.com

Introduction

Any art student can draw fur or grass with just a few simple strokes, but this is still a significant obstacle in real-time graphics. This work attempts to adapt the techniques of traditional artists to 3D in order to achieve stylized representations of furry objects at real-time frame rates.

What's a Graftal?

A graftal is a small "fin" of geometry that is used to represent a single piece of fur or other complex geometry. Each graftal has individual parameters such as size, orientation, style, etc.



Graftals are represented by a central spine (red) and an array of rib lengths (blue). Each frame, the graftals are expanded by their rib vectors into fins.

This expansion process is more expensive than simply storing static geometry, but it allows for complex animation techniques such as wind and gravity. It also makes it easy to render the graftal in different styles and levels of detail (e.g. only draw the spine)

Graftal Creation

Currently, Graftal normals and positions are based on the vertices of the original model *(Fig B*). This works for models of significant triangulation, but may produce grid-like noise (visible in *Fig D). Fig C* shows the result of this basic setup.

Artists' work (and our world) is never perfect, so the uniform graftals method appears unnatural (*Fig C)*. To alleviate this, we perturb several graftal parameters when they are initialized, such as: size, normal, and rib width (*Fig D*).

Level-Of-Detail (LOD)

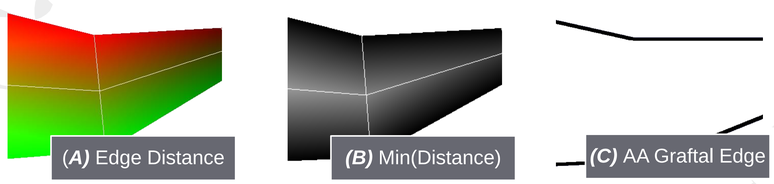
As the graftals travel through the rendering pipeline, they are represented as a cloud of point-vertices (visualize *Fig B)* with many attributes. These attributes, such as visibility, distance, and orientation, are used to select a discrete Level-Of-Detail (LOD) for the graftal. This allows for different styles, such as the outer and inner graftals seen in *Fig F.*

In order to save processing time, the cloud is first passed through a culling stage, where graftals that will not be drawn are discarded (such as those near the center of the bunny in *Fig F)*.

Temporal Coherence

Individual images often look pleasant, but stylized techniques often break down when viewed in an interactive scene, as the image appears to "pop" between frames. To alleviate this, we scale the size of the graftals based on their distance from the edge of the object. Thus graftals from behind appear to "grow" over the horizon, instead of simply popping in *(Fig E).*

Edge Rendering



When the central spine is expanded to form the graftal geometry, the expansion distance (from each edge) is stored at each new edge vertex *(A)*. In the pixel shader, we threshold this value to determine if we are near an edge *(B)*. If so, we draw an edge line *(C)*. If it is near the outer edge of the geometry, it is blended with the background to anti-alias (AA) the line.

Results

Stages	Time (ms)	# Graftals	Time/Graftal
Culling	0.3	70,000	~
LOD Sort #1	0.1	35,000	~
LOD Sort #2	0.1	35,000	~
LOD Draw #1	4.7	10,000	470 ns
LOD Draw #2	1.3	2,700	481 ns
Total:	**6.5 ms**	**70,000**	**93 ns**

Timings for *Fig F.* The results show that the system, given a budget of 16ms (which is robust!), should be able to handle about 170,00 graftals. For reference, a typical video game character will have ~50,000 graftals (1 graftal per vertex).

Future Work

- **Perturbed graftal positions:** Instead of using the object's vertex positions, which causes grid noise
- **Grass-style graftals:** To showcase flexibility
- **Speed improvements:** There is still much optimization that can be done, especially in the shading and culling stages
- **Graftal animation:** such as wind or gravity

Fig F: Final Result

Fig A: Diffuse Model

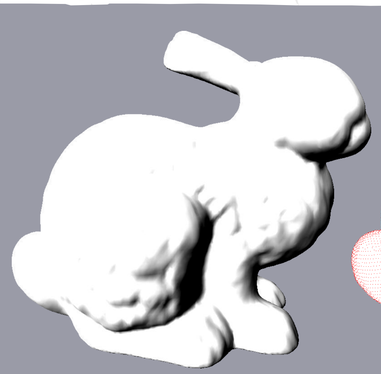


Fig B: Graftal Positions

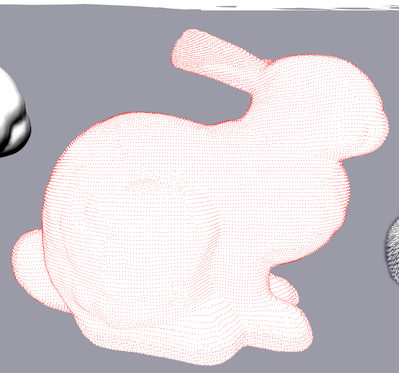


Fig C: Uniform Graftals

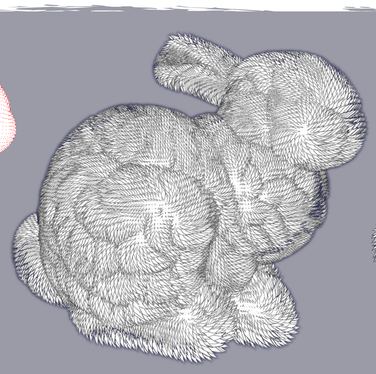


Fig D: Perturbed Graftals

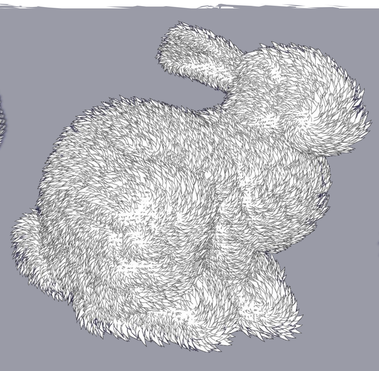


Fig E: Distance Scaling