Thern, the Nano Technology from John Carter's Mars

Richard Pickler[*] Cinesite
Simon Stanley Clamp[†] Cinesite
Artemis Oikonomopoulou[‡] Cinesite

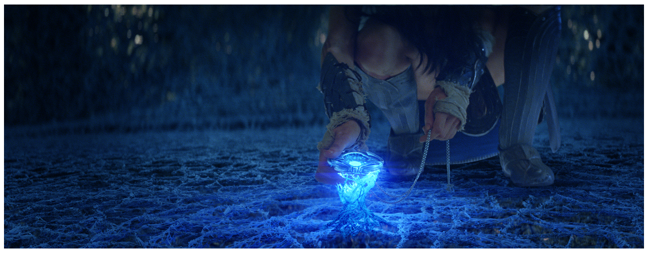

Figure 1: *Close encounter with Thern*

1 Introduction

Thern, or the Ninth Ray, as it was referred to in John Carter, covered a broad scope of related effects throughout the film. Twenty different variations fell under the umbrella of Thern, ranging from small standalone assets, to entire rooms filling the screen, and to effects that were only vaguely recognizable as Thern. Along with a wide range of concept art and reference footage of natural phenomenon, we keyed on a few conceptual ideas from the director. From "somewhere between organic and mechanical" we focused on a vegetation growth technique [Palubicki et al. 2009] with modifications specific to our needs. From "fractal", we derived a recursive technique that we could direct between different levels depending on how we wanted the viewer to interpret the image. Animation grew out of disparate references and the ingenuity of the team, culminating in a creative use of a very old algorithm.

2 Look

Early concept work at Cinesite produced a static model of a small section of Thern. While this intricate prototype was hand modeled and textured with traditional techniques, it was never an option for the large scale set elements. L-Systems proved too difficult to direct into the complicated structures needed, and a Voronoi based system was rejected because the results were not visually organic and far too regular. Ultimately, we developed a system based upon vegetation techniques entirely within Houdini. We implemented the , the biggest challenge came when we had to make the growth loop back upon itself. We accomplished this by turning the branch-avoidance method into branch-attraction gradually over time.

Our initial brief on the structure described it as a fractal. Specifically, if we were to zoom in enough, the same structure should emerge to the viewer. Our original concept attempted this through an intricate texture, which proved limiting in both artists' time and our animation abilities. Instead, we adopted a recursive concept directly in our modeling workflow. We modeled the gross structure through one level of simulation, then after setting relative sizes of individual branches and some manual massaging, we filled that structure with the very same tool. Conceptually, the first simulation is the larger structure the audience perceives, and the fill step is what it was actually made of. When rendering this mass of curves, however, the initial images were noisy and the higher level structure was lost. We borrowed from concepts used at other studios [Petrovic et al. 2003] to manipulate the normal of the high frequency geometry in order to bring out the higher level shapes of the model. The lighters could then balance between the details and structure as each shot required.

3 Animation

In an effort to maintain flexibility in our system, no matter how we generated the geometry, we kept a strict division between the generation of the geometry and the animation. Despite the generation happening through a growth algorithm, it would be difficult to direct both visually pleasing geometry and an appropriate animation at the same time. Therefore, we chose to animate our geometry based on values created from a simple Dijkstra Algorithm [Dijkstra 1959]. This allowed us to completely change where the geometry grew from and groom the speed of animation, significantly improving our reaction time to director requests.

Our Dijkstra implementation allowed us to select the starting point we wanted to grow from, and based upon the cost along each edge, determine at what time every line segment would animate. We could vary the speed of each curve segment to guide the growth into particular shapes, or even to produce subtle patterns in the leading edges. The speed at which all segments would grow was expressed in frames, and the Dijkstra algorithm would produce a rank for every vertex which defined what frames it was animating on. Those values were then used to define when the geometry moved. If the values of the two vertices on a segment were greater than the current frame, the geometry was simply deleted. If they were less, the geometry was static. If the values straddled the current time, the frame was normalized between the values and then fed through a library of canned animation styles. These values were also passed through to rendering, and fed into various shading parameters to focus the eye on the leading edge of animation and blend back into the static volume.

4 Acknowledgements

The full development of Thern happened with a great team building upon each other's work. None of this would have been possible without the creativity, collaboration, and hard work of everyone involved.

References

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 269–271. 10.1007/BF01386390.

PALUBICKI, W., HOREL, K., LONGAY, S., RUNIONS, A., LANE, B., MĚCH, R., AND PRUSINKIEWICZ, P. 2009. Self-organizing tree models for image synthesis. In *ACM SIGGRAPH 2009 papers*, ACM, New York, NY, USA, SIGGRAPH '09, 58:1–58:10.

PETROVIC, L., HENNE, M., AND ANDERSON, J. 2003. Volumetric methods for simulation and rendering of hair. *Simulation*, 06-08, 1–6.

[*]e-mail: rpickler@gmail.com
[†]e-mail: ssc@cinesite.co.uk
[‡]e-mail: artemis@cinesite.co.uk

SIGGRAPH 2012, Los Angeles, California, August 5 – 9, 2012.
ISBN 978-1-4503-1435-0/12/0008