Underwater Procedural Vegetation on Pixar's *Luca*

Marlena Fecho
marlena@pixar.com
Pixar Animation Studios
USA

Brennan Mitchell
brennanm@pixar.com
Pixar Animation Studios
USA

Jamie Williams
jam@pixar.com
Pixar Animation Studios
USA

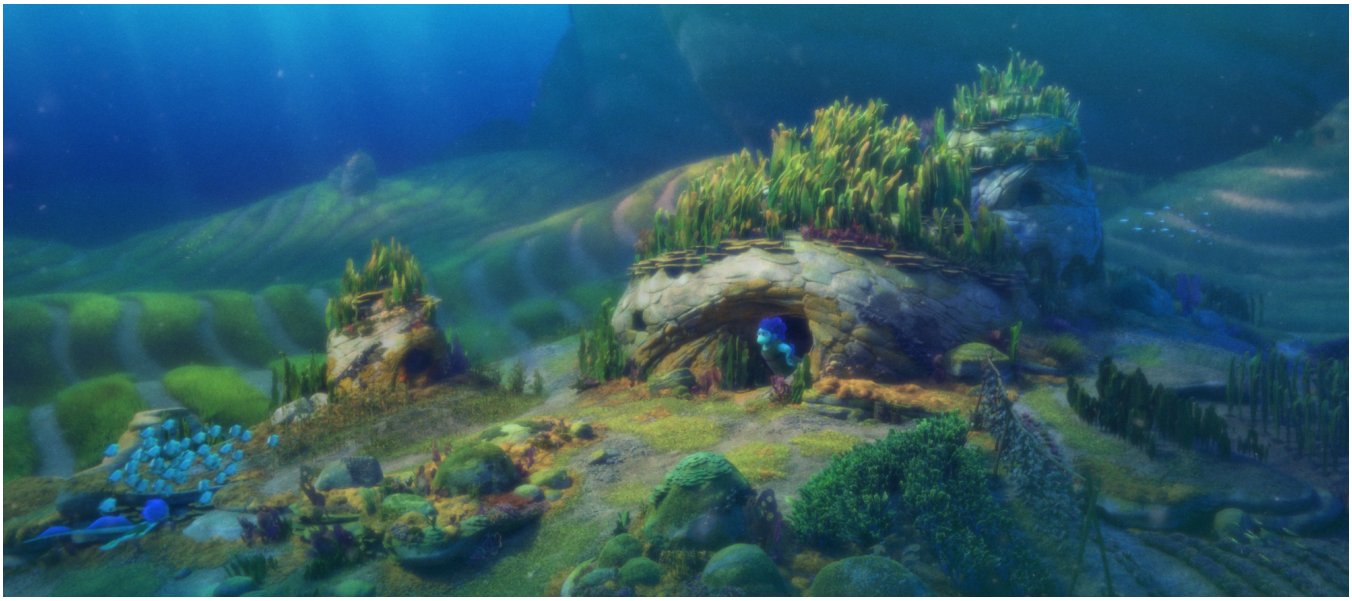

Figure 1: In the depths below *Luca's* Portorosso, lush greenery is always in motion. ©Disney/Pixar

ABSTRACT

On *Luca*, we extended the procedural vegetation and debris system, called Moss, which has been in use at Pixar since *Brave*, with capabilities to create lush underwater seascapes. The Moss system was modernized for *Onward* to be OSL (Open Shading Language) based, enabling artists without C++ expertise to develop new Moss types. For *Luca*, we added a deformable mesh geometry type for complex underwater vegetation. Additionally we added SeaWind, an OSL-based procedural motion module, to hit the specific art direction of flowing curves in underwater currents. We also improved the pipeline which brings procedural geometry into Houdini for integrating the hero simulation with the procedural motion seamlessly.

ACM Reference Format:
Marlena Fecho, Brennan Mitchell, and Jamie Williams. 2021. Underwater Procedural Vegetation on Pixar's *Luca*. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '21 Talks), August 09-13, 2021.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3450623.3464671

SIGGRAPH '21 Talks, August 09-13, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8373-8/21/08.
https://doi.org/10.1145/3450623.3464671

1 BACKGROUND

The Moss system is used to procedurally create ground cover, like grass of many varieties, pebbles, barnacles, and other debris. It is responsible for a lot of the vegetation on a film, often in the background, so it must be efficient and directable with broad strokes and bulk edits.

Geometry is generated at render time based on a core system written in C++, which defines canonical types that can be generated, such as simple curves, hierarchical curves and instances. The specific geometry of a Moss asset is defined by a shader written in OSL. [Dixon et al. 2020]. The OSL shader is typically customized for the specific needs of the film. These Moss assets are attached to ground geometry, and can be controlled with parameters and painted masks for things like density, size, and clumping.

The nature of motion underwater is that neighboring things are pushed in the same direction by the currents. Also, production design on *Luca* was very interested in the appeal of slow, flowing S-curves. The existing core system provided procedural keep-alive motion, but this subsystem was designed to emulate the motion of grass moving in air. Its motion was fairly stiff and had a high degree of randomness, which was not appropriate for the underwater environments.

Sometimes character interaction is needed, like for foot (or fin) contact. For those cases, we have a system to capture procedural

geometry, and export it to USD (Universal Scene Description), so that hero simulation work can be done on it in Houdini.

2 PROCEDURAL GEOMETRY

A number of changes were made to the geometry capabilities of the Moss system. For one, in order to achieve the desired S-curve shapes, we generalized the existing curve type to support an arbitrary number of control vertices. Additionally, the existing mesh type was instance-based, and thus non-deformable. So to support more complex shapes that could also move with the underwater currents, we added a new canonical deformable mesh type. To assist creating Moss assets of this new type, we wrote a script to export geometry data from Maya into a format that could be consumed and modified in OSL.

To support better art direction of the procedural geometry, we forwarded additional signals such as ground surface UVs to the generated geometry, which could be used to drive shading across the terrain for specific art notes.

With these new tools in hand, we were able to create Moss assets such as seagrass, seaweed, mushroom coral, and anemones.

3 PROCEDURAL MOTION

We introduced a new keep-alive module, specifically for underwater motion, called SeaWind. Underwater, plants move slowly and coherently with the ebb and flow of the water. Even a gentle current will push everything in the same direction. Given the global nature of the motion, we chose to use a shared module to drive the effect of the SeaWind applied to all Moss assets, to ensure that everything appeared to move in the same currents.

Because the look of the SeaWind was specific to the art direction of *Luca*, we wrote it in OSL instead of adding it to the C++ core. This also allowed rapid iteration and fast deployment.

SeaWind computes a per-vertex offset, making it applicable to both curves and meshes. It is driven by a 4D noise function, where the noise is sampled at the root of the generated element (blade of grass, leaf of seaweed, etc.) and at the current time. The tip of the element uses the same noise position as the root, but with some time delay, which is controlled by a "stiffness" parameter. A fully stiff element would be pushed in the same direction at the root and tip, but a low-stiffness one would be pushed at the root first, and at the tip later, creating a flowing ripple as the current changes direction. The resulting offset is also scaled by the normalized distance from the root; i.e. the root is fixed and the tip moves the most. An additional length-preserving step was needed to prevent the SeaWind from stretching or squashing the geometry.

The technical artists controlled SeaWind motion with global parameters for speed and spatial frequency of all Moss assets. They also had per-asset parameters to adjust amplitude, stiffness and per-blade randomness (flutter). We also found we needed a way to reduce motion near walls and other large objects to reduce obvious collisions, so we added a paint-driven control to dampen the SeaWind amplitude.

4 HERO SIMULATION

Because Moss is a render-time procedural system, by default geometry is not available upstream of rendering. But if a character needs to interact with it, we have a workflow called VegCapture, originally created for *Onward*, to bake out per-frame Moss geometry to USD so that the simulation artists can import it and integrate it into their simulation work. VegCapture is able to extract the geometry in a targeted region around a character. The captured regions are also removed from the procedural version at render-time to avoid duplicate geometry when the simulated version is reinserted.

Using VegCapture, we import curve-type Moss geometry into Houdini, where we use a Vellum solver to compute high quality animation that is based on collisions with the animated characters.

In the existing system, a pre-pass would extract a rest pose from the per-frame data and store offsets to represent the procedural motion. The simulation would then run on the rest pose and simulate interactions with the characters. Finally, the offsets of the procedural motion would be added back in on top of the simulation.

For underwater procedural motion, however, this was not sufficient because these offsets were often large and could move the simulated geometry back into a colliding state. We therefore extended the Houdini simulation to update the rest pose with the procedural animated data at each frame, and to use springs to pull the curves back to the procedural motion while respecting the collision results of the Vellum simulation.

As an optimization, we used collision detection against the character to limit which curves needed to be simulated. We also added controls to blend between fully procedural and fully simulated motion, which helped the results integrate naturally even with dramatic procedural motion.

5 CONCLUSIONS AND FUTURE WORK

The Moss system wasn't originally designed to have procedural keep-alive that is stylized per film. On *Luca* we added our SeaWind procedural motion to the OSL-based geometry creation phase. While not well integrated in the existing C++ core, it proved to be highly flexible and allowed us to hit the art direction on this film. It would be useful to formalize this concept and have a dedicated OSL pass for custom keep-alive motion. More generally, it could be powerful to modularize the OSL components for generating, modifying and deforming geometry, and allowing artists to assemble them in a node graph, leading to more reuse and extending the accessibility to less technical artists.

We also encountered a fundamental difficulty in sharing the procedural motion logic between the OSL system and the Houdini-based simulator. It would be beneficial if, in the future, one implementation could be shared between these systems.

ACKNOWLEDGMENTS

Many thanks to Matt Benson, Francisco De La Torre, Antony Carysforth, Mike Ravella, Andy Whittock, Dave Dixon, Inigo Quilez, and the many other people who have contributed to the Moss architecture and design over the years.

REFERENCES

Dave Dixon, Matt Johnson, Andy Whittock, Peter Roe, Jamie Hecker, and Matt Kuruc. 2020. Procedural Geometry with Open Shading Language on Pixar's Onward and Soul. In *ACM SIGGRAPH 2020 Talks* (Virtual Event, USA) *(SIGGRAPH '20)*. Association for Computing Machinery, New York, NY, USA, Article 38, 2 pages. https://doi.org/10.1145/3388767.3407372