Art-Directed Surface Tearing Simulation

Leon Jeong Wook Park*
Pixar Animation Studios
Pixar Technical Memo #19-01

Figure 1: The first and last frames of body suit tearing scenes with simple lighting setup (c) *Disney/Pixar all rights reserved.*

A key sequence in Pixar's *Cars 3* includes dramatic surface tearing effects on *Lightning McQueen*'s new body suit. Controlling and stylizing physically-based simulation is one of the hardest topics in CGI productions. To achieve our director's specific vision, we developed an art-directable surface tearing simulation framework. This talk presents our detailed implementation and framework for modeling, simulating, and post-processing the torn surface.

Pre-Simulation

We designed and implemented a palette of tools for geometry pre-processing. It involves intuitive, artist-friendly geometry cutting tools as well as procedural geometry fracturing processors based on mathematical models.

We had very specific art directions from directors. The locations and the shapes of tearing had to be 100% controllable. First, we implemented paint and/or sketch based geometry cutters. Users can generate tearing topology intuitively by painting and/or sketching on the given surfaces.

We also implemented procedural geometry cutting tools. This tool set self-defines fractured topology. It operates with a special attribute map called 'stress map'. Users can provide painted stress maps. Also, this tool can voluntarily generate the stress map by analyzing input geometry topology such as curvature, polygon complexity, correlation between normal and wind vector, as well as by assuming possible collision in current scene.

The last step of pre-simulation is generating the initial simulation attributes map. A set of key simulation parameters, such as rest geometry attraction, constraint strength, deformability, restitution force, and stress are intuitively paintable by users. we also provide functionality to generate the parameter set procedurally based on input geometry's velocity, topology, scene complexity, and given stress map.

The pre-processed geometry keeps interpolated rest shading reference position attributes so that it can be rendered in the desired texture coordination.

Simulation

We implemented a multi-solver structure composed of 3 sub-solvers in order. The first solver deforms the input rest geometry. The deformation is a combination of custom bending and distortion where high stress values are detected. Specifically, the solver adds extra details on the high-stress edge area. The rest geometry attracts the main simulation geometry so we can design specific shapes during the simulation. This idea was proposed to implement peeling-like effects on the leading torn area.

The next process is the main finite element simulation. We isolate the region of interest and only simulate that. If a scene requires tearing multiple regions, we can split the simulation by region and parallelize it. Later all the simulated and non-simulated surface regions will be combined with boundary interpolation.

Then, the last solver modifies the input stress map. This solver predicts potential high-stress areas during current simulation step and updates the stress map with newly calculated and accumulated stress values by analyzing acceleration. We can turn on and off that process. In fact, the new updated stress map suggested by the solver conducts simulation to get more natural dynamics in general.

Post-Simulation

Often times, we should clean up, reshape, or filter simulation output to produce stable, coherent geometry. We designed art-directable and procedural post-process tools to reduce iterating simulation.

We implemented functionality to mix multiple simulation caches. It enables blending multiple caches with art-directed bias map. Also, layering caches is supported with user-defined weight per each layer.

Another very important process was filtering the simulation cache. Sometimes the simulation output has jittery areas. Instead of applying filters to whole geometry, we apply filters to only complex areas. We adopted the entropy concept to represent complexity and discontinuity per point.

$$S = -k\sum_i p_i \log 2(p_i)$$

where p is the distance from current point's position to the mean position of k-nearest neighbors. (We can also expand p with any other attributes than position, such as velocity, if we want to generate entropy for velocity.) In thermodynamic entropy, k is the Boltzmann constant. In our case, however, we designated $1/\pi^2$ as constant k. The filter amount is varied per entropy S. The higher entropy areas will be more filtered. With this technique, we can keep beautiful high frequency details while the jittery area is filtered out. We also implemented a sketch-based filtering tool so that users can define the areas which will be force-filtered.

The last process is merging very close points so that we can get clean topology. Thus, final geometry topology is time-varying. We used Pixar's Universal Scene Description (USD) to exchange the custom topology-varying geometry between applications.

*e-mail: coolgauss@gmail.com