Procedural Approach to Animation Driven Effects for Avengers: Endgame

Tobias Mack
tmack@wetafx.co.nz
Weta Digital
Wellington, New Zealand

Ashraf Ghoniem*
ashraf.ghoniem@gmail.com
Blizzard Entertainment
United States of America

Ruben Mayor
rmayor@wetafx.co.nz
Weta Digital
Wellington, New Zealand

Gerardo Aguilera
gaguilera@wetafx.co.nz
Weta Digital
Wellington, New Zealand

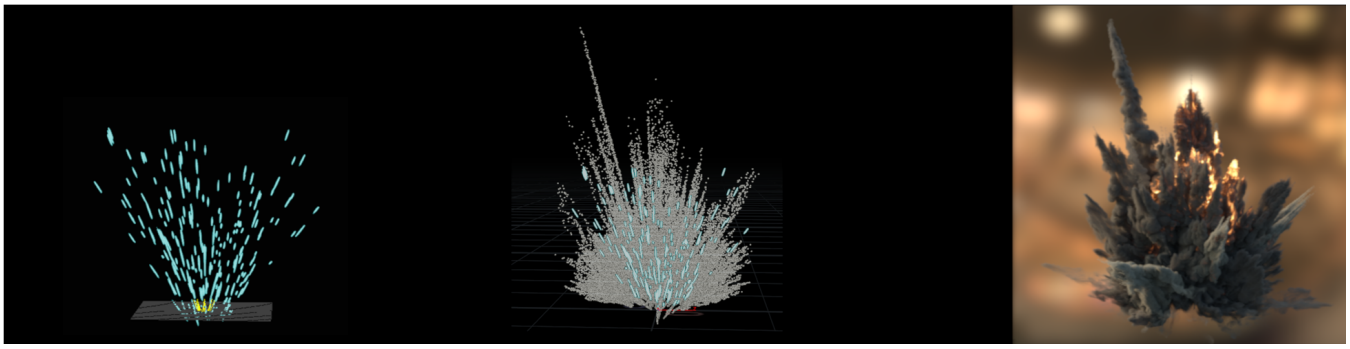

Figure 1: Animation in Maya (left) FX in Houdini (middle) Final render (right)

ABSTRACT

This article describes the workflow for delivering real-time FX elements to animation in an omni-directional way and how they get processed for further stages, as it was used on 'Avengers: Endgame' at Weta Digital. We will discuss our procedural approach to solving some of the challenges around what we call 'AnimFx', focusing on production requirements and our ensuing technological solution.

CCS CONCEPTS

• **Computing methodologies** → **Computer graphics**.

KEYWORDS

procedural pipeline, FX, real-time animation, automation

ACM Reference Format:
Tobias Mack, Ashraf Ghoniem, Ruben Mayor, and Gerardo Aguilera. 2019. Procedural Approach to Animation Driven Effects for Avengers: Endgame. In *Proceedings of SIGGRAPH '19 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3306307.3328184

*worked at Weta Digital during the production of 'Avengers:Endgame'

SIGGRAPH '19 Talks, July 28 - August 01, 2019, Los Angeles, CA, USA

ACM ISBN 978-1-4503-6317-4/19/07.
https://doi.org/10.1145/3306307.3328184

1 PRODUCTION

'Avengers: Endgame' presented a unique challenge for the FX department at Weta Digital. It demanded a high volume of pure FX work combined with tasks mainly influenced by direct animation - like tracer fire, thrusters, squibs, explosions, and missile trails - all in a very condensed timeframe, with a desire to reduce the human hours required.

To do this, we looked at the shots and tried to automate repetitive tasks so most of them could be set up procedurally. After the FX look development process, the result of the motion shot work could directly flow into compositing. This freed up artist time and allowed the data to be processed automatically for use in downstream departments. As part of

this, we tried to break up the traditional 'assembly line' approach by using procedural assets - reducing or eliminating smaller FX tasks completely.

In situations where the data returned back to the FX team, we wanted to make sure that they could rely on the data and visualization being exactly the same as the animator and the client (for animation approval) saw them. This was very important to exchange timing and placement information consistently across multiple departments.

As well as data integrity being a concern, we also wanted to reduce kickbacks from animation about publishes that weren't physically plausible. Providing animation with real-time feedback on behavioral limits through the animation rig reduced the number of iterations required, saved time for the artists and let us reach a more believable result for the client faster.

2 TECHNOLOGY

The AnimFx workflow is based on our in-house real-time evaluation engine Koru and Weta's real-time renderer, Gazebo. We chose this after analyzing several products - looking at existing in-house and external solutions such as the Houdini Engine. The decision was mainly driven by the real-time requirements for animation rigs and that both software packages were already integrated in most of our hosts throughout our pipeline.

Koru is a LLVM-based, data oriented all-purpose evaluation engine that offers GPU capability to the user building the session in it. It has Gazebo already integrated, allowing us to bypass the hosts completely and directly evaluate data outside the host then draw it back into the viewport with the other elements in the scene. This made it the perfect candidate for the AnimFx workflow as we had to ensure data and visualization consistency across multiple hosts and satisfy the real-time performance needs of motion capture and animation setups. Having full control over the software components enabled us to make the work-flow highly procedural and focus on a cacheless approach. In Maya, Koru gets exposed through a MpxNode to pass data from the host to the engine. For the Houdini integration we exposed Koru through a custom packed primitive, enabling us to expose data selectively to the user. This gained performance in Houdini and meant the visualizer allowed us to draw directly into our viewport integration through Gazebo. Koru was already successfully used for animation rigs on previous productions such as War for the Planet of the Apes, so the next logical step was to extend its use into the FX workflow connecting to animation.

The goal was to build a library of basic, generic 'building blocks' within Koru. With these, the FX artist can configure and develop the look of the asset in Houdini, experiencing the same look and behavior as the animator and later the client approving it. All building blocks are designed with stability and performance in mind over flexibility. The 'ballistic' building block can be used for a multitude of cases and is - despite being a simulation - fully scrubbable without having to cache data to disk. The artist can modify parameters like gravity or initial velocity dynamically in real-time without playing back from the beginning. We do this by using data oriented structures allowing for fast runtimes which can recalculate the whole iteration every frame, thereby saving artist time.

Most building blocks also use a system called 'Governor'. This is a part of the Koru graph that enables the FX artist to limit the controls for animation so the data coming back is suitable for physically plausible simulations or other use cases in FX. The 'Governor' gives the animator real-time feedback in their scene on overstepping the physical bounds such as flight speed, sideways movement or target control. This reduces kickbacks from FX for physically implausible or undesirable elements that cannot be used to drive the high-resolution simulation.

Once the asset is configured in Houdini, an animation rig is automatically built and published with the Houdini graph using the same building blocks. Animation only publish the curves to the rig, which can then be imported back into Houdini and replayed to recreate the data without having to load any cache. This eliminates animation having to publish complex data. The FX artists can then either use the re-evaluated data to drive simulations, or use it for timing reference. The AnimFx system also has some automation built in, so it is possible to run automated conversion processes for data passed on to later pipeline stages. For a higher degree of proceduralism the workflow also allows the use of a render procedural such as Houdini Engine or other in-house procedurals to deliver data to lighting.

3 FUTURE WORK

Future integration of the evaluation engine with Nuke would enable us to directly deliver elements to later stages in the pipeline. This could be particularly interesting for elements that don't need to go to animation but could be directly created in compositing.

Enabling dynamic real-time representation of FX assets in animation will allow us to produce many different elements and tools ranging from magic effects, real-time deformation previews to more complex water approximations.

ACKNOWLEDGMENTS

We like to also acknowledge the work Jamie McCarter has done for the project. He was an essential member of the team.