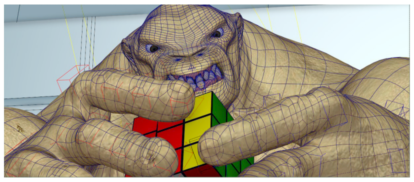
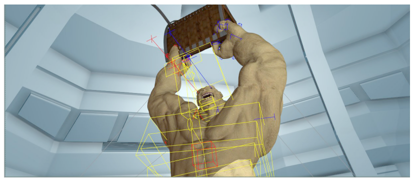
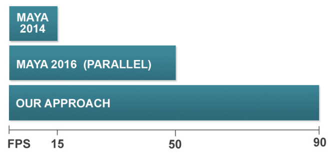
A Flexible Rigging Framework for VFX and Feature Animation

Jesus R. Nieto Theo Facey Sylvain Brugnot
Double Negative[*]

Left, middle: previs for "Rubix", an internal animated short. Right: performance comparison for a standard rig

Abstract

We present the work recently undertaken to ensure that Pinocchio, our long-standing, production-proven modular rigging platform, can meet the demands of both VFX and Animated Features. By transitioning to a fully modular, DCC-agnostic framework we enable rigs to be built across a number of host applications and leverage a variety of evaluation back-ends; this in turn allows us to target different stages in our pipeline more efficiently, by offering high performance to animators and easy prototyping to riggers, while retaining the ability to share assets with other companies if needed. Furthermore this approach involves comparatively small upfront development and artist training costs.

Keywords: animation, rigging, DCC-agnostic, Fabric, KL

Concepts: •**Computing methodologies** → **Animation;**

1 Introduction

Today's VFX companies are required to produce ever-greater numbers of increasingly complex CG characters in ever-shorter time frames. In Marvel's Avengers: Age of Ultron, Ultron's rig comprised thousands of moving parts, for a total of over 16 million polygons. Animated features, on the other hand, often use simpler rigs but can have many more of them: Pixar's Monsters University, for example, featured over 400 characters.

To meet these demands, some of the larger Feature Animation companies have built their own standalone animation platforms [Watt et al. 2014], which offer great performance and are tailored to the artists needs. Unfortunately, the substantial development and training costs required put this approach beyond the reach of most other companies, and although they result in highly optimised rigs, the long prototyping periods involved are impractical for the shorter time frames of VFX and Broadcast.

Standard animation packages like Maya, on the other hand, enable more agile rigging workflows but have only just begun to address this explosion in complexity. As an example, a typical production-quality character rig in Maya 2014 contains over 3000 nodes and runs no faster than 15fps, resulting in a frustrating experience for riggers and animators alike. While the recent introduction of multi-threaded evaluation has gone a long way towards alleviating performance issues, there is still room for improvement.

2 Our Approach

Our in-house modular rigging system, Pinocchio, lets the user create rigs by building and connecting individual modules (eg, 'arm', 'leg', 'spine'). Whereas these were previously defined in Maya-specific terms and stored as MEL scripts, we now use a DCC-agnostic representation to encode the rig as a combination of 'rig units'. A Python translation layer is then responsible for turning this high-level description into a set of low-level building instructions in the host application.

In the same manner, the algorithms used in our rigs had originally been implemented using the Maya C++ API, creating a dependency we were keen to remove. We opted instead for Fabric Software's KL language, which offers a good compromise between portability, ease of coding and performance. While the KL integration in Maya provided by Fabric is very flexible, in order to achieve maximum performance we provide our own integration in the form of highly optimised C++ wrapper nodes, which are procedurally generated in order to reduce development times.

By implementing different translation layers it therefore becomes possible to leverage the strengths of different platforms and computational back-ends: within Maya, the same rigs can be built using standard nodes for maximum compatibility when outsourcing, fully editable Fabric nodes when rig prototyping, and highly optimised but opaque custom KL/C++ nodes for animating. We are also investigating how this mechanism can be used to deploy these rigs in Houdini for simulation purposes or in Riot, our in-house crowd system.

3 Results

By expressing rigs in non-DCC-specific terms, the new framework allows riggers to focus more on broader rig behaviour and less on implementation details, which results in simpler rig graphs and faster prototyping. Using a combination of KL and C++ enables us to offer animators clear performance gains over standard Maya rigs while keeping development costs low. Overall, the flexible nature of the system lets us easily tailor the rig implementation to specific usage scenarios, allowing us to take on the challenges of both VFX and Feature Animation.

References

Watt, M., Coumans, E., ElKoura, G., Henderson, R., Kraemer, M., Lait, J., and Reinders, J. 2014. *Multithreading for Visual Effects*, 1st ed. A. K. Peters, Ltd., Natick, MA, USA.

[*]e-mail:{crn,tdf,sylv}@dneg.com

SIGGRAPH '16, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4282-7/16/07
DOI: http://dx.doi.org/10.1145/2897839.2927463