Optimizing Rig Manipulation with GPU and Parallel Evaluation

John Kahwaty
Walt Disney Animation Studios

Walter Yoder
Walt Disney Animation Studios

Andy Lin
Walt Disney Animation Studios

Gene S. Lee
Walt Disney Animation Studios

David Suroviec
Walt Disney Animation Studios

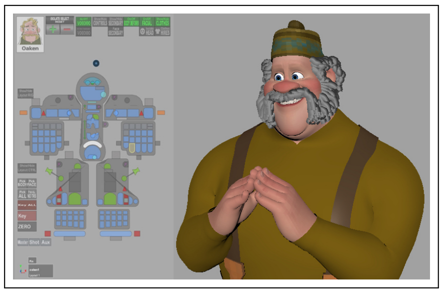
Fig 1: *Example Production Rig*

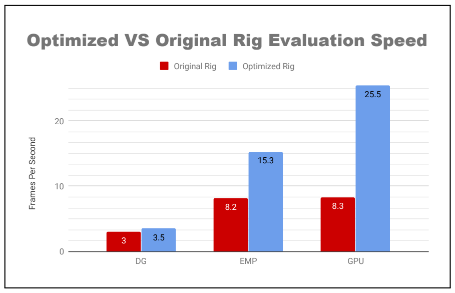

Fig 2: *Rig Performance in DG/EMP/GPU*

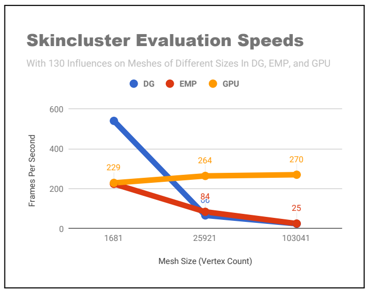

Fig 3: *Scalability with DG/EMP/GPU*

ABSTRACT

Rig speed plays a critical role in animation pipelines. Real-time performance provides instant feedback to artists, thereby allowing quick iterations and ultimately leading to better quality animation. A complete approach to real-time performance requires both playback and manipulation at interactive speeds. A pose-based caching system (PBCS) addresses the former, but the manipulation of complex rigs remains slow. This paper speeds up rig manipulation by taking advantage of modern multi-core architectures and the GPU, and by constructing rigs that evaluate efficiently on parallel processing hardware. This complete approach, including tool updates and rig optimizations, was used successfully to significantly improve interactive rig manipulation performance on *Frozen 2*.

CCS CONCEPTS

• **Computer systems organization** → **Real-time systems**; **Real-time system specification**.

KEYWORDS

Real-Time Interaction, GPU, Parallel Evaluation, Pose-Based Caching

ACM Reference Format:
John Kahwaty, Walter Yoder, Andy Lin, Gene S. Lee, and David Suroviec. 2019. Optimizing Rig Manipulation with GPU and Parallel Evaluation. In *Proceedings of SIGGRAPH '19 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3306307.3328181

1 INTRODUCTION

In key-framed animation, animators manipulate hundreds of rig controls while reviewing the motions repeatedly. The speed at which animators perform these tasks influences their ability to iterate and ultimately create high quality animation. PBCS [Lin et al. 2015] is an effective tool for improving the review process, however it is not suitable for interactive manipulation because rig control adjustments generate new poses and therefore diminish the utility of caching geometry.

Modern animation systems evaluate scene graphs using multiple threads. However, parallel evaluation alone is insufficient to guarantee real-time interactive performance. Critical tool updates and rig optimizations are often required to ensure optimal system performance. The remainder of this paper discusses the process of tuning our software and rigs to achieve significant performance improvements by leveraging Maya's Evaluation Manager in Parallel mode (EMP) [Autodesk 2016], coupled with GPU deformers.

2 APPROACH

2.1 Tool Updates for Parallel Evaluation

To take advantage of the EMP, it was necessary to update existing tools to run safely and correctly in multi-threaded environments.

2.1.1 Make Deformers Thread Safe. Every custom deformer was upgraded to be thread safe for multi-threaded execution. The OpenMP library was replaced with Intel's TBB library to avoid thread oversubscription. Plus, special instrumentation was added to deformers to troubleshoot performance issues using Maya's profiler.

2.1.2 Break Up Complex Deformer Nodes. Many of our deformers combined multiple functions into a single node in order to take advantage of internal parallelization within a single-threaded evaluation environment. However, this embedded parallelization creates a bottleneck in the EMP. Breaking up these complex custom deformer nodes into smaller, simpler nodes enables the EMP to schedule the deformers in parallel and speed up evaluation.

2.1.3 Convert Python Nodes to C++ Nodes. Python's global interpreter lock (GIL) forces all python nodes to be scheduled serially. This prevents EMP from evaluating efficiently and causes a major bottleneck in system performance. Therefore, custom Python nodes in the rigs were converted to C++.

2.1.4 Create A Custom Evaluator for PBCS. The traditional DG uses a pull evaluation model while the EMP uses a push evaluation model. With DG, PBCS skips costly rig evaluations by not pulling output geometries. With EMP, PBCS must decide if costly rig evaluations can be skipped at the beginning of an evaluation cycle. This is accomplished by creating a custom evaluator that skips nodes upon cache hits and thereby shortens the time of rig evaluations.

2.1.5 Implement GPU Deformers. All custom rig deformers were re-implemented to work with the GPU. The kernels were written in OpenCL which allows the EMP to deform geometry on the GPU and pass the output data directly to a viewport renderer.

2.2 Rig Optimizations

The following rig optimizations focus on procedural ways to post process rigs to make them suitable for multi-threaded evaluation and GPU acceleration. Each was designed to enhance performance while preserving a workflow designed to optimize the authoring process.

2.2.1 Create Unique Deformer Chains. Costly deformer nodes that affected multiple meshes were split into several deformers, with each ultimately affecting only one mesh. This reduced the dependencies between meshes and allowed each mesh to be evaluated separately. It also enabled each mesh to be deformed by a unique chain of deformers, which is required to utilize GPU acceleration. This is the most impactful rig optimization we made.

2.2.2 Optimize Mesh Size. Large meshes are costly to evaluate and often create a bottleneck, where many idle threads wait for a few busy ones to finish. To avoid this situation, we tried cutting large meshes into a collection of smaller ones. On average, cutting the body of a rig into eleven pieces and the face into six, with approximately 1000 to 2000 vertices per mesh, produced the greatest performance gains in EMP. The cut-up meshes were recombined into a single mesh via a custom parallel blending node.

Cutting large meshes improved EMP performance, but also increased GPU mesh copy overhead. The number of meshes sent to the GPU has a greater impact on performance than the size of the meshes themselves. Experimental results showed that the GPU operated most efficiently with 20 to 50 meshes when running on 24-core machines. Since the GPU gains far exceeded any improvements we were seeing in EMP performance in terms of scalability, ultimately we combined geometry to lower the number of meshes sent to the GPU.

2.2.3 Break Large Evaluation Graph Cycles. In EMP, a large cycle in the evaluation graph creates a bottleneck. All nodes in an evaluation graph cycle are grouped into a single cluster and executed by the same thread. For example, all nodes in an inverse kinematics (IK) system naturally form a cycle. If a rig depends on an IK system to evaluate before it is scheduled, most of the rig is evaluated by just a few threads. Some cycles can be removed by using alternative rigging methods. For instance, some automation can be replaced with efficient pose-space deformers.

2.2.4 Remove Output Mesh Dependencies. Any deformation that depended on the evaluation of another output mesh was modified to depend on a smaller intermediate mesh, with a duplicated deformation chain. This eliminated the slowness incurred by evaluating the other output mesh and allowed the GPU to evaluate the deformation chains in parallel.

2.2.5 Remove Head Geometry. Historically, the head geometry was cut off and blended back onto the body to optimize single-threaded performance. This allowed the face rig to exist on its own mesh and to be turned off completely when disabled. However, this setup prevented a large portion of the rig from being sent to the GPU. To remedy this, all head deformers were transferred to the body, making it possible to remove the head from the rigs entirely.

3 RESULTS

The interactive manipulation performance of the optimized character rigs was measured in terms of DG, EMP, and GPU evaluation modes. Figure 2 contains performance comparisons with and without rig optimization for the character rig shown in Figure 1. The benefits of using parallel evaluation can be seen even in the original rig which shows a 2.7x speed-up in EMP when compared to the single-threaded DG performance. However, the original rig was not structured for GPU deformation, and therefore did not obtain any performance gains with GPU deformations enabled. With the optimized rig, EMP provided 4.4x speed-up over DG, and enabling the GPU provided an additional 1.7x speedup over EMP alone. This shows that rig optimizations are necessary to take full advantage of parallel evaluation and GPU deformations.

As discussed in Section 3, the number and size of the meshes has a significant impact on performance. Furthermore, optimizing for parallel performance is counter to optimizing for GPU deformation. In order to investigate the scalability of the GPU to accelerate deformations, isolated tests were conducted on a single mesh to examine the effect of mesh resolution on each evaluation mode (Fig 3). As the vertex count goes up, traditional DG evaluation as well as EMP performance falls quickly while, in contrast, the GPU performance scaled better as mesh complexity increased. It is worth noting that with low vertex counts, the DG evaluation can perform better than both EMP and GPU because of the associated overhead. These tests show that the GPU has a much higher potential to scale as characters become more complex.

4 CONCLUSION

Rig manipulation performance was significantly improved using Maya's EMP with GPU-accelerated deformations. Tool updates were necessary to leverage the parallel and GPU evaluation, and rigs were optimized for the new parallel evaluation model. These adjustments were applied to most characters on *Frozen 2*, resulting in an 8x increase in interactive performance on average, with many characters achieving real-time manipulation rates. By updating tools and rigs to utilize parallel and GPU evaluation, we have opened up the potential to take advantage of future hardware advancements, such as additional cores and more powerful GPUs.

REFERENCES

Autodesk. 2016. Maya. (2016). https://www.autodesk.com/products/maya
Andy Lin, Gene S. Lee, Joe Longson, Jay Steele, Evan Goldberg, and Rastko Stefanovic. 2015. Achieving Real-time Playback with Production Rigs. In *ACM SIGGRAPH 2015 Talks (SIGGRAPH '15)*. ACM, New York, NY, USA, Article 11, 1 pages. https://doi.org/10.1145/2775280.2792519