LibEE 2 - Rich Authoring and Fast Evaluation

Stuart Bryson
DreamWorks
stuart.bryson@dreamworks.com

Esteban Papp*
DreamWorks
esteban.papp@gmail.com

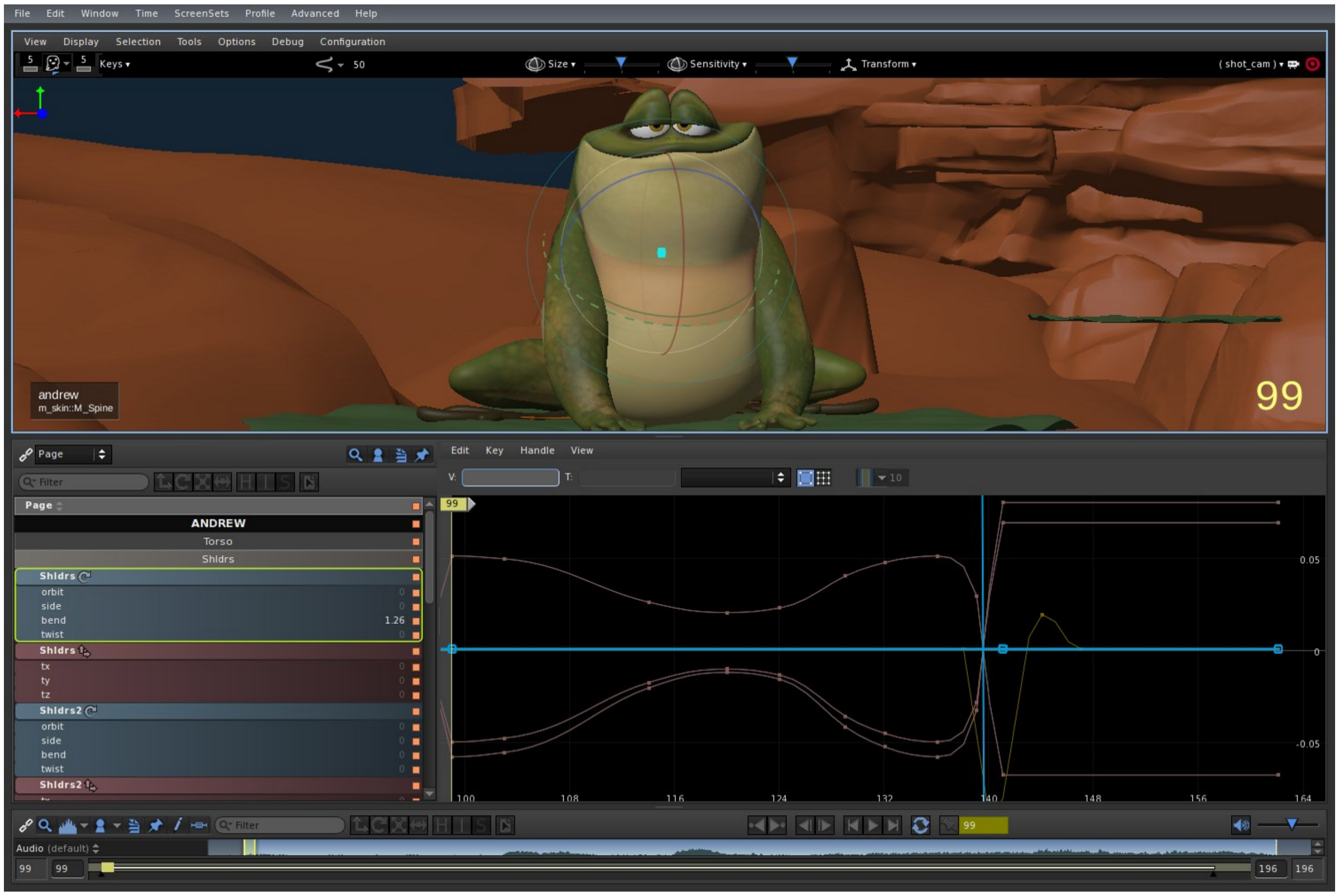

Figure 1: Premo (Property of DreamWorks)

CCS CONCEPTS

• **Computing methodologies** → *Parallel algorithms*; *Animation*;

KEYWORDS

directed acyclic graph, dependency graph, evaluation, authoring, animation, rigging

*Now at Amazon.

ACM Reference Format:
Stuart Bryson and Esteban Papp. 2018. LibEE 2 - Rich Authoring and Fast Evaluation. In *Proceedings of SIGGRAPH*. ACM, New York, NY, USA, 2 pages.

1 INTRODUCTION

The Premo animation platform [Gong et al. 2014] developed by DreamWorks utilized LibEE v1 [Watt et al. 2012] for high performance graph evaluation. The animator experience required fast evaluation, but did not require fast authoring or editing of the graph. LibEE v1, therefore, was never designed to support efficient edits. This talk presents an overview of how we developed LibEE v2 to enable fast authoring of character rigs while still maintaining or improving upon the speed of evaluation. Overall, LibEEv2 achieves a 100x speedup of authoring operations compared with LibEE v1.

2 PREVIOUS AUTHORING APPROACH

Previously, character rigs were authored in a proprietary tool called Rig. These rigs were translated offline into a LibEE v1 graph. This graph was then loaded into Premo and evaluated. Since most editing operations were not supported in Premo, if a change to the rig was required, the process must repeat. Only minor graph editing operations were allowed within Premo, such as loading and unloading assets or editing of constraints. Therefore, during development of LibEE v1, evaluation performance was our priority, not authoring.

LibEE v1's performance was not obtained from walking the graph topology itself, but through a series of caches and lookups that were built during the first evaluation. For example, one cache tracked the relationship between animator controls and output geometry. When the graph structure changed, LibEE v1 needed to rebuild these caches.

Since the set of inputs and outputs for an animation rig is fairly static, and editing the graph in an animation session was fairly limited, LibEE v1 was sufficient. Character riggers, however, do not have a static set of inputs or outputs. They are constantly modifying and inspecting the results at arbitrary points in the graph. LibEE v1 was not sufficient for rigging workflows. LibEE needed improvement to support editing without compromising evaluation performance.

The LibEE v1 caches that enabled the fast evaluation did not scale for typical Character Rigging workflows. Caches only stored lookups from animation controls to output geometry. Character Rigging workflows have orders of magnitude more inputs and outputs that are not required for animation workflows. Caching this amount of data was prohibitively expensive and took significantly longer to rebuild.

Another limitation was the translated graph did not match the original authored semantics. Features such as subgraphs and references were all flattened during translation from Rig. LibEE v1 did not have the ability to represent these capabilities.

3 LIBEE V2 AUTHORING APPROACH

LibEE v2 takes a different approach. While many concepts were borrowed from the original implementation, it is an entire rewrite. LibEE v2 creates a dual representation of the graph: one for authoring, and one for evaluation. The authoring representation has Nodes, Attributes, and Connections, while the evaluation representation has Tasks and Dependencies. As the Character Rigger makes an edit to the authoring representation, we simultaneously update the evaluation representation. This allows us to support a rich authoring language while maintaining fast evaluation.

During the update to the evaluation representation, we can apply various optimizations. Firstly we remove duplicate task dependencies, we create shortcuts that bypass connections between two inputs, and also create inlined task functions that remove the need for an extra node to convert between types.

LibEE v1 was slow at performing graph walking operations used in dirty propagation and planning. It did these in serial and relied on caches to reduce the need for walking operations. In LibEE v2, however, the evaluation representation reduces the number of steps required to perform a walk, and also performs them in parallel.

LibEE v2 adds a shared memory pool. This memory pool decreases memory consumption by allowing values to be reused rather than duplicated. It supports multidimensional data caching, allowing inputs and outputs to be cached across frames.

4 RESULTS

LibEE v2 edits the graph faster, evaluates subsequent results faster, improves performance and reduces memory consumption as compared to LibEE v1. When adding a node that has no connections, LibEE v1 would take 239ms to edit the graph and 170ms for a subsequent evaluation. This is compared with 1ms and 7ms respectively for LibEE v2. When setting a value that is not an animator input, LibEE v1 would take 168ms to edit the graph and 143ms for a subsequent evaluation. LibEE v2 would take less than 1ms and 6ms respectively (See figure 2). When changing which outputs we were evaluating, LibEE v1 would take 184ms compared with 11ms with LibEE v2. Lastly, a typical character in LibEE v1 used 1.464 GB compared with 1.126 GB in LibEE v2.

All these improvements were achieved while maintaining and often improving the speed of evaluation. For example, on one of our production characters we went from 15fps in LibEE v1 to 24fps in LibEE v2 while posing the main body control.

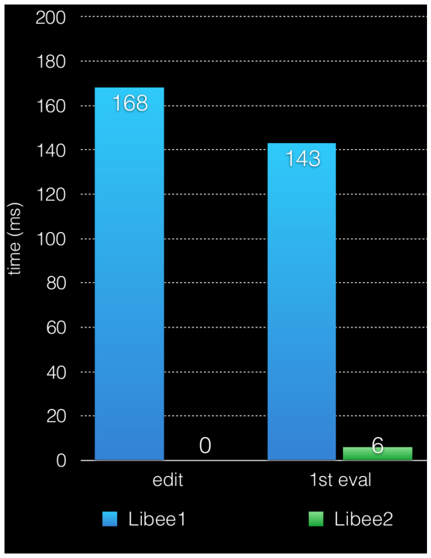

Figure 2: Setting a value that is not an animator input

REFERENCES

Matthew Gong, Fredrik Nilsson, Alex Powell, Jason Reisig, Alex Wells, Stuart Bryson, Esteban Papp, and Paul DiLorenzo. 2014. Premo: A Natural-Interaction Animation Platform. In *ACM SIGGRAPH 2014 Talks*. ACM, 3.

Martin Watt, Lawrence D. Cutler, Alex Powell, Brendan Duncan, Michael Hutchinson, and Kevin Ochs. 2012. LibEE: A Multithreaded Dependency Graph for Character Animation. In *Proceedings of the Digital Production Symposium (DigiPro '12)*. ACM, New York, NY, USA, 59–66. DOI: http://dx.doi.org/10.1145/2370919.2370930