Handling Scene Constraints for Pose-Based Caching

Gene S. Lee
Walt Disney Animation

Christian Eisenacher
Walt Disney Animation

Andy Lin
Walt Disney Animation

Noel Villegas
Walt Disney Animation

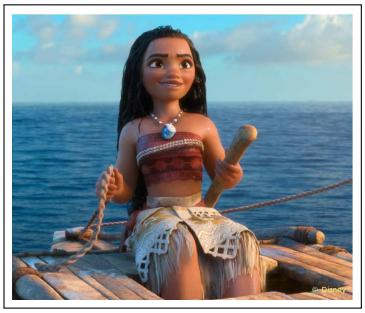

Fig 1: *Moana constrained to boat, oar, and rope*

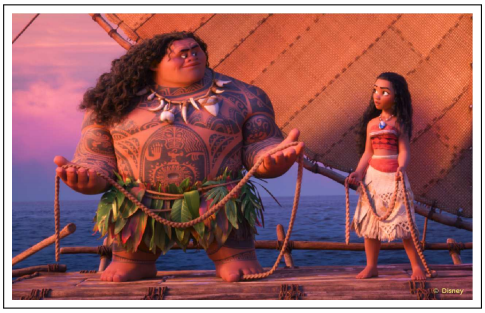

Fig 2: *Constraints position Maui's and Moana's hand*

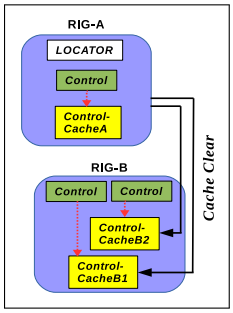

Fig 3: *Sample invalidation graph*

ABSTRACT

This paper presents a conservative, uniform method for handling scene constraints, such as *look at* and *parent*, in a pose-based caching system. The constraints are organized into a dependency graph where nodes represent the control caches of a rig, and directed arcs link the rigs to specific control caches. For any animation update, the dependency graph indicates which cache to clear and which other rigs to subsequently update. This method supports pre-evaluation, avoids expensive state tracking, and is easy to implement. The result is a seamless experience that works with all types of constraints while preserving real-time performance.

CCS CONCEPTS

• **Computer systems organization** → *Real-time system spec*;

KEYWORDS

Constraints, Invalidation Graph (InvGraph), Pose-Based Caching

ACM Reference format:
Gene S. Lee, Christian Eisenacher, Andy Lin, and Noel Villegas. 2017. Handling Scene Constraints for Pose-Based Caching. In *Proceedings of SIGGRAPH '17 Talks, Los Angeles, CA, USA, Jul 30 - Aug 03, 2017,* 2 pages.
https://doi.org/10.1145/3084363.3085047

1 INTRODUCTION

In a pose-based caching system (PBCS), such as [Lin et al. 2015], real-time playback is achieved by caching both geometry and pose data. An initial drawing pass stores the geometry according to its pose and subsequent drawing passes use the same pose to efficiently retrieve and display the cached geometry. The pose itself is also cached to avoid repetitive, and generally costly, evaluation of rig controls. The pose data store every rig control into a *RC-cache*, which is invalidated by animation updates.

1.1 Problem

The proper upkeep of RC-caches becomes complex with the introduction of scene constraints, which relate the rig controls or states of multiple rigs. These constraints may be animated and are uniquely configured per scene. For example, in Figures 1 and 2, *parent* constraints bind the characters' body to the deck of the boat, and their hands to either a rope or an oar. Any movement of the boat or the two props will also move their hands.

Scene constraints introduce complexity by altering the mapping of rig controls to RC-caches. Normally bijective, the mapping now extends across multiple rigs. One control can map to multiple RC-caches, and similarly, an RC-cache can map to several controls. With this added complexity, it is more difficult to determine which RC-cache to clear after specific rig controls are updated.

The problem compounds for constraints that relate the state of one rig to another. In this case it can be difficult to determine which controls influence the final pose. For example, in Figure 2, if Maui's gaze is bound to Moana's face, any number of pose controls could affect the state. The corresponding RC-cache for Maui's gaze is influenced by a potentially great number of pose controls.

2 SOLUTION

Our solution is a conservative, effective method of handling scene constraints for PBCS. Constraints in a scene are organized into an *invalidation graph* (InvGraph) – a single dependency graph with nodes for each rig and directed arcs for each constraint. The arcs relate rigs, not rig controls, to RC-caches. This design links the update of a whole rig to the clearing of specific RC-caches.

Figure 3 presents a sample InvGraph. Each node consists of a set of controls, state variables, and RC-caches. The dotted arrows inside a node identify the mapping of controls to RC-caches. The directed arcs between nodes express constraints, each clearing a different RC-cache. Any time RIG-A is updated, based on changes to its input rig controls, both of the RC-caches of RIG-B will be cleared, and RIG-B will be subsequently updated.

This solution has the advantage that it works for all types of constraints (e.g. parent or orientation), and inputs (e.g. pose control

SIGGRAPH '17 Talks, Jul 30 - Aug 03, 2017, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5008-2/17/07…$15.00
https://doi.org/10.1145/3084363.3085047

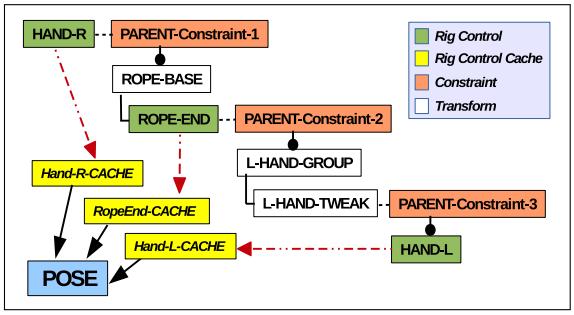

Fig 4: *An Object Hierarchy with Three Parent Constraints*

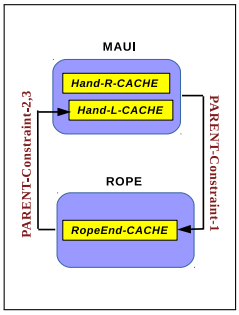

Fig 5: *Invalidation Graph for Fig 4*

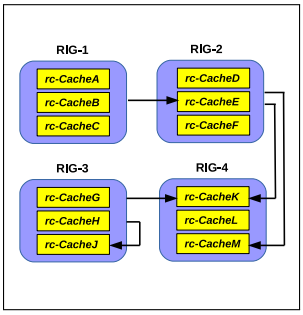

Fig 6: *Complex Invalidation Graph*

or state). Every constraint and input is uniformly identified by the rig that is the driving target and by the cache that is constrained.

2.1 Invalidation Graph

Figures 4, 5, and 6 provide concrete examples of the construction and use of the InvGraph. Figure 4 visualizes the connections and object hierarchies of three parent constraints (PC-1, PC-2, PC-3) from Figure 2. Each parent constraint allows objects in separate hierarchies to form a temporary parent-child relationship. PC-1 tightly parents the base of the rope to Maui's right hand, while PC-2 and PC-3 loosely parent the end of the rope to his left hand. The tweak transform between the two constraints permits the left hand to be pulled off the rope. The red arrows show the bijective mapping of the rig controls to their corresponding RC-caches.

Figure 5 shows the InvGraph for the object hierarchy of Figure 4. The two nodes identify the RC-caches of Maui and the rope. The directed arc, linking the nodes to RC-caches, are set by searching the graph for the first rig controls that exist upstream and downstream from the parent constraint. For PC-1, the upstream control is HAND-R and the downstream control is ROPE-END-CACHE. PC-2 and PC-3 share the same upstream and downstream node. Hence, both constraints map to a single directed arc.

Figure 6 shows a more complex InvGraph. Four rigs are bound by five varying types of constraints, one of which starts and ends at RIG-3. Self-referencing arcs occur when a constraint links one rig part to another, such as the clasping of hands. With more constraints and nodes, a single rig control update can cascade into a series of RC-cache clears and multiple rig updates. For instance, an update to RIG-1 induces a cache clear of RIG-2, which forces RIG-2 to update, which in turn induces RC-cache clears of RIG-4.

3 RESULTS

Our solution has several benefits. First, the InvGraph is easy to implement. All arcs originate at a node, independent of the rig control from which they originate. Cycle detection is performed with nodes only, independent of the number of rig controls and RC-caches involved. Second, the simplicity of our solution eliminates the odds of clearing the wrong cache. Invalid RC-cache updates are troubling for PBCS and are often difficult to debug.

Third, the cache invalidation process does not involve the tracking of state, such as the position of a surface locator. State tracking is problematic in a PBCS when multiple background processes (BGs) are working in parallel to fill the cache. Augmenting these parallel processes to communicate both pose and tracking data is expensive. Forcing the primary process to evaluate the results of every BG leads to a performance bottleneck. Our approach simply assumes that every potentially trackable state of a node changes when any of its rig controls update.

And finally, prior to any rig control update, the results of updating a rig can be fully pre-computed. All downstream cache clears are discovered efficiently via forward propagation. For instance, in Figure 6, the pre-computed caches cleared for RIG-1 are RC-caches E, K, and M. For RIG-3, the caches cleared are RC-caches J and K.

3.1 Assessment

One limitation of our system is that it can be overly conservative. It may unnecessarily clear caches that are still valid. To remedy this situation, one could replace the InvGraph with a complete dependency graph that precisely links rig controls to pertinent RC-caches. The graph could either be integrated with the animation dependency graph or maintained separately. Either way, the complete graph must be traversed via forward propagation to find the most efficient set of RC-cache to clear and rigs to reset.

However, building a complete dependency graph requires greater effort. First, the setup may not be agnostic of constraint type. Each new type of constraint can potentially add a different set of connections. Second, it is necessary to set up state tracking callbacks which are expensive when multiple cache-filling, background processes are involved. Third, it can be inefficient to pre-compute the effects of every rig control change. When multiple constraints are involved, the complexity can grow combinatorially.

In practice, our conservative approach has not proven to be a hindrance to performance. Artists normally animate with a modest number of scene constraints, so the extra cache clears are rarely excessive. We chose the benefits of simplicity and uniformity over a truly optimal approach which would entail more overhead and introduce greater complexity to the system.

3.2 Application

The InvGraph was developed for our PBCS during the making of "Moana" and successfully tested on many shots from the film. It has been deployed in our production pipeline and is in active use. With this new functionality, scene constraints are created dynamically and applied seamlessly in our pose-based caching system while maintaining real-time performance.

REFERENCES

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. DOI: http://dx.doi.org/10.1145/2775280.2792519