Wrangling Lighting and Rendering Data at Disney Feature Animation

Mark A. McLaughlin *Mark.McLaughlin@disney.com*
Joseph M. Lohmar *Joe.Lohmar@disney.com*
Ernest J. Petti *Ernest.Petti@disney.com*
Lewis Wakeland *Lewis.Wakeland@disney.com*
Chris Springfield *Chris.Springfield@disney.com*

Walt Disney Feature Animation

1 Introduction

The amount of data that can comprise a shot in an animated CG feature film is tremendous. All of this data must be assembled into a scene by our Shot Finaling department, where lighting and final rendering takes place. The data arrives in many formats and at different stages of completion, necessitating a system that can assemble new scenes and update existing scenes easily. In addition, the intricate manner in which various parts of the scene relate to each other varies with the action being performed and must be tracked. We present a user-friendly system for wrangling scene data and relationships, which serves as the foundation for our internal lighting package, ***Lumiere***, currently in use in production on *Chicken Little*.

2 System Overview

Data in a scene is broken down and organized by types, such as cameras, lights, materials, geometry, bindings, etc. Related items of the same type are grouped together and stored in ***data containers***. Containers also maintain the methods and information needed to import, export, update and render their data. Data containers are controlled by ***data container managers*** which manage all containers of the same type. Every data container manager is tracked by the ***Data Container Manager Registry***. An element in a scene (say a character) is comprised of a collection of data containers, which encapsulate the element's geometry, materials and bindings. The ***Relationship Manager*** handles associations between these containers and also knows how each element is constructed and how to import the appropriate containers for the element. The ***Render Manager*** is used to render scenes by causing the necessary containers to export their data and invoking the render action. The system has been implemented in Alias' Maya using the C++ API for speed, with a corresponding MEL interface.

3 Data Organization Details

The data container is the basic unit of organization. All managed scene data is stored in data containers. Data containers are implemented as Maya dependency nodes. Although a data container can be any type of dependency node, most are object sets. The system is able to incorporate "outside" packages (such as our fur/hair system) via methods that transform the existing data into containers so the two systems can coexist. Information related to a data container is stored in attributes of the data container node and associations between containers are defined via attribute connections. In this way the data management organizational information is persistent. Since all data containers share a common interface (attributes and methods), new data types can be easily added to the system. This provides for future expansion as additional needs arise.

All of the data containers of the same type are controlled by a single data container manager. Container managers are implemented as Maya object sets with associated information stored in attributes of the set nodes. Each manager has a method for importing a data file of the proper type and placing the imported data into a container. The data in a container can also be swapped out with other data. Each data container manager has a *default* container which is automatically created and used to store newly created (hence unmanaged) data. There are methods for converting existing scenes into "managed" scenes by automatically sorting out all scene data and placing it into default containers. Methods also exist for converting scenes from any previous version of the system to the current version.

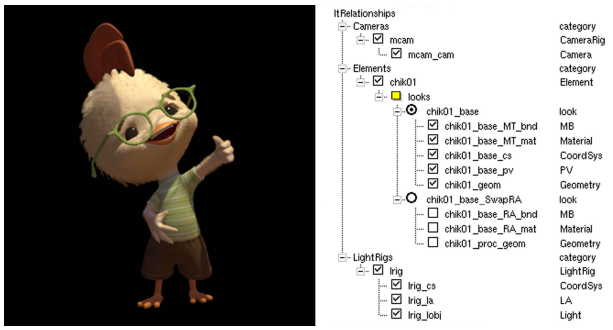
The Data Container Manager Registry keeps track of all of the existing data container managers. It can be queried for a list of current managers which in turn can be queried for lists of existing containers. Thus all managed data can be accessed via the registry.

All managed data is tracked by the system and any change to the data will cause its container to become "dirty". This will also notify all other associated containers of the change. Only dirty containers need to be exported (to transfer the data or render the scene), improving the efficiency of the system. Similarly, if a container is "locked", its data is not exported for rendering; rather a preexisting render data file is used. This allows the data in the live Maya scene to be decoupled from the data used for rendering, making it possible, for instance, for the Maya scene to contain light-weight proxy geometry.

4 The Relationship Manager

The Relationship Manager is an abstract layer above the data containers which builds a hierarchical view of the scene from a configuration file and defines groups and their associated containers. This view of the data management system gives the artists the ability to switch between different looks and easily turn elements on or off for rendering. The user can interact directly with the Relationship Manager's graph or use a selection mechanism which allows them to choose by name, group type, or data container type.

The figure below shows the hierarchy of a relationship view for a scene and the corresponding rendered image. Checked objects represent "enabled" containers (which are visible in the render) and unchecked objects represent "disabled" containers that still exist in the scene. Unchecking at the element level disables all the child containers. Switching the look for an element would turn off all the containers associated with the old look and turn on all the containers for the new look. There are also a set of MEL commands that allow users to process the groups on the Maya command line or through other scripts.



Special thanks to Dean Edmonds of Gooroos Software for his contribution to this project.