Can't See The Jungle For The Trees

Tom Melson*
MPC

Figure 1: *Jungle Life. ©2016 Walt Disney Pictures. All rights reserved.*

Abstract

For Disney's *The Jungle Book* we needed to build huge and highly detailed jungle sets. The scenery consisted of thousands of different models, placed millions of times. This requirement, together with the modelling intricacy of vegetation, meant there was a lot of data to handle both in terms of numbers of instances and overall polygon count. We needed a tool that could manage and display this amount of data in real time. With more than 100 distinct set builds, used in over a thousand shots, efficiency was all important.

Keywords: Environments, Preview, Set Building

Concepts: •**Information systems** → **Multimedia information systems;** •**Computing methodologies** → *Computer graphics;*

1 An API for Building Environments

The ModelHierarchyManager (MHM) is our in-house tool that allows us to create, edit and display our set builds. The individual models themselves (trees, plants, rocks etc.) come from the Assets department. It is the job of the Layout and Environments departments to assemble the sets from these to produce the "Model Hierarchies" that will represent our environments. The MHM is now the default tool for doing this, building on work done in 2010 for Prince of Persia [Meeres-Young et al. 2010].

*e-mail:tom-m@moving-picture.com

SIGGRAPH '16 Talks, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4282-7/16/07
DOI: http://dx.doi.org/10.1145/2897839.2927429

The tool has been built in a modular way. At its core there's a Python API that allows the loading of assets and the construction of the composite structures. These structures are essentially just assets that contain a number of child assets and an associated list of transform matrices for each, to represent the position of each instance. By separating the geometry from the layout transformations in this way we allow layout changes to be made without necessarily needing to load any of the underlying model data.

The API provides methods to load, assemble and edit the structures as well as a lot of extra functionality to aid in building and querying scenes. Aside from the main tool, the API is also used directly by various other parts of the pipeline and in bespoke workflows to read and create Model Hierarchies.

The MHM has a UI that can be run as stand-alone or within a DCC like Maya. It allows scenes to be loaded, asset structures to be manipulated and for various procedural editing to be applied. It can also provide statistics on asset numbers, Level Of Detail (LOD) etc.

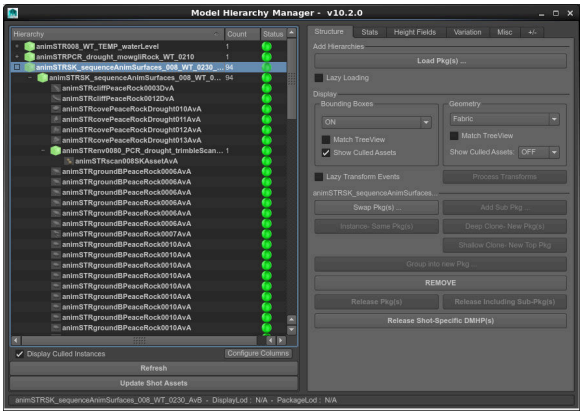

Figure 2: *ModelHierarchyManager UI*

2 Visualisation

To visualise the environment and to manually edit a layout, the tool is run inside of Maya. There is a plug-in style system for the display, where different "Display Managers" can be selected to present the scene in different ways. The default option represents the scene hierarchy using nested Maya transform nodes, each with a shape node for the bounding box. However, the Maya scene is built lazily and the hierarchy only constructed as and when the user asks for the respective part to be displayed. The entire hierarchy can be navigated in the MHM UI and selected sections chosen to be visualised, either in bounding box form or to load the actual geometry. Originally this would mean importing the model's Maya asset but to increase performance we gave the option to use Fabric Engine's InlineDrawing GL engine to draw directly to the viewport [Fabric Engine 2016]. Using Fabric Splice nodes allowed the geometry to be loaded much more quickly and memory efficiently. Also the display rate is maximised, effectively bound only by the GPU. The tool makes use of the different LODs built for assets, defaulting to the lowest available and there's also an option to use proxy geometry when available (a very low resolution model used only as a place-holder in Maya).

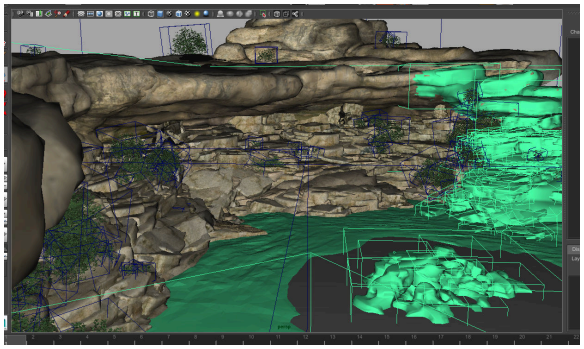

Figure 3: *ModelHierarchyManager Use In Maya. ©2016 Walt Disney Pictures. All rights reserved.*

As the builds on *The Jungle Book* got larger and the numbers of instances increased, the performance of the MHM Maya display began to suffer. For example some scenes might include a scattering of leaves and debris which could constitute hundreds of thousands of instances. Even with the geometry handled as efficiently as possible, the instance count meant that the sheer number of Maya transform nodes and connections could become unmanageable. To alleviate this we created an alternative Display Manager that is 100% Fabric. We accessed KL classes from our Python code using Fabric "RT Vals". The entire scene hierarchy is represented in Fabric KL and the display, including bounding-boxes etc. is handled in Fabric, again using InlineDrawing. With large scenes this gave us a speed improvement for setting up and displaying the scene in Maya of 50 times or more. Initially this was a read only display mode but we implemented a ray-casting based selection system in KL, then used a proxy Maya bounding box and Maya events to allow the user to select and edit the scene contents directly in the viewport.

3 Shot Specific Editing

MPC builds its main environment assets in a central place, with many shots then referencing the same generic assets. But we found a need to be able to override them in this shot level context. Firstly specific instances might need to be moved or removed, but also many shots may not actually need to include most of the set at all, as the camera might focus only on one specific area.

There was already work under way to create a new "Dynamic Model Hierarchy" asset, which wraps a Model Hierarchy and allows dynamics to be run on the geometry, such as wind through the jungle trees. We decided to extend the use of these assets and make them able to override attributes of instances in a layout, changing the LOD, culling or transforming them. This includes an automated process of culling and LOD assignment based on the shot camera's frustum. Whenever an underlying asset changes, a process will fire, which uses the MHM API to interrogate the camera and instances. The instance bounding boxes are processed and any outside the frustum (plus an additional tolerance) are marked as culled. The bounding box is also used to calculate a rough size on screen for each instance and this is used to assign an LOD based on some pre-set ranges. These pre-sets can be configured for the show, then optionally overridden on a scene or shot basis. There's also an option to add overrides based on the asset name. For example we could create a rule so that anything with "Rock" in the name will never pick the lowest LOD. These rules can also be defined on the show, scene or shot level.

Doing all this up-front, before rendering meant we could load the assets into Maya and get the benefits of the automated LODs and culling at the layout stage, significantly improving efficiency. Any manual changes then made by the artist were taken as additional shot specific overrides and are also written into the Dynamic Model Hierarchy, all without affecting the original underlying asset.

We also provided functionality to allow copying of shot-specific changes between contexts. For example an artist may have made some changes for a shot, moving some rocks and culling some bushes that were not required. It might then be decided that these changes are needed on other shots or maybe across the entire sequence. They could do this with a few clicks from within the UI. Alternatively it was also possible to transfer shot-specific changes onto the original asset, which would then be inherited by all shots. When copying this data, the option was given to include any or all of the LOD, culling or transformation overrides.

4 Use for Presentation and Reference

The tool was fully integrated into the MPC pipeline and has become functionally very rich. Using Fabric gave us a huge performance advantage and also allowed us to load the asset textures and add effects like fog, making viewport renders an appealing and efficient option for making presentation dailies. There was also a read-only version of the MHM used by the Animation, Crowd and FX departments to keep their scenes up to date with the latest environments.

Building the tool in a modular way like this really paid off, with the API, UI and various display modes proving useful at many places throughout the pipeline and in many film departments.

Acknowledgements

Clair Bellens and Robert Tovell for their work on the DMHP pipeline. To Ivan Castane Capel, for the support with Fabric Splice and KL. And Dora Morolli for the technical feedback.

References

FABRIC ENGINE, 2016. http://www.fabricengine.com/.

MEERES-YOUNG, G., RICKLEFS, H., AND TOVELL, R. 2010. Managing thousands of assets for the city of alamut. In *SIGGRAPH Talks*.