Forging a New Animation Pipeline with USD

Aloys Baillet
Animal Logic
aloysb@al.com.au

Eoin Murphy
Animal Logic
eoinm@al.com.au

Oliver Dunn
Animal Logic
oliverd@al.com.au

Miguel Gao
Animal Logic
miguelg@al.com.au

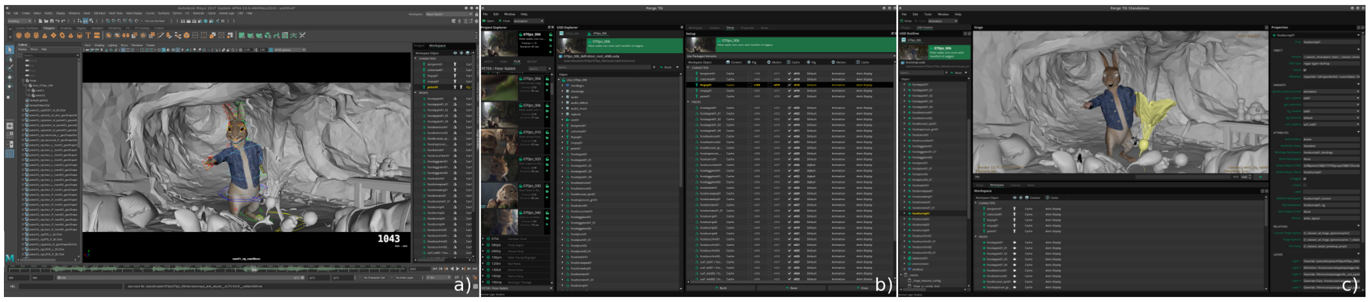


Figure 1: a) *Forge Animation* in Maya®; b) *Forge TD* in Maya®; c) *Forge TD Standalone*

ABSTRACT

The *Peter Rabbit* movie features 5 hero characters and dozens of secondary characters animated across more than 1100 shots.

We introduce some practical and production-proven solutions to integrate Pixar™ Universal Scene Description (USD) into Autodesk Maya® based on our now opensourced *AL_USDMaya* plugin, and how they were used to create a high performance and intuitive animation platform.

CCS CONCEPTS

• **Computing methodologies** → **Animation**; • **Human-centered computing** → *User centered design*;

KEYWORDS

animation pipeline USD maya

ACM Reference Format:
Aloys Baillet, Eoin Murphy, Oliver Dunn, and Miguel Gao. 2018. Forging a New Animation Pipeline with USD. In *Proceedings of SIGGRAPH '18 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3214745.3214779

1 INTRODUCTION

In early 2016, we decided to use Autodesk Maya® for our rigging system as it was the most optimal system for high performance rigs; however we also knew it would not be able to scale to the levels of complexity required by our typical productions in terms of amount of geometry and viewport playback speed for non-rigged elements.

We initially considered embedding our in-house scene description into Maya® for elements of the scene such as environments and geometry caches of characters and props. Indeed, we knew our own scene description could efficiently represent huge amounts of data thanks to its nested instancing system relied upon by our proprietary renderer Glimpse, but we also knew it was lacking more advanced features such as layering and variants.

Around that time, Pixar™ announced the opensourcing of their own scene description called USD, which they kindly gave us early access to. We soon decided to adopt USD and to build a more advanced bridge into Maya®, which is now fully open-sourced and called *AL_USDMaya.*

We created a proprietary *Forge* software stack to generate our USD files and to present both simple and more advanced views to users. *Forge* also allows to load these files both outside and inside of Maya®.

2 ANIMATION SCENE INTEGRATION IN MAYA®

Forge relies on the *AL_USDMaya* bridge, whose primary objective is to make USD the owner of the whole scene data, directly streamed by Pixar's Hydra GL renderer into the Maya® viewport for optimal performance.

Our bridge listens to USD state changes and selectively translates some elements of the USD scene into corresponding Maya® objects. Such objects are mostly cameras (Maya® does need a native camera to have a functioning viewport) and animation rigs loaded as Maya® References.

A plugin system allows custom translators to be registered against specific USD Prim types. Among a few other opensourced custom types, the ALMayaReference translator creates a Maya® reference node on activation, unloads the reference when the prim becomes inactive, and automatically updates the reference path when a different USD variant is selected. This live tracking of the scene allows us to express different rig levels of details as USD variants.

3 ANIMATION SCENE WORKFLOW

In *Forge*, Animators are presented with a list of their assigned shots for which they can choose to load all the latest approved shot data from read-only geometry caches with a single click. In order to animate the characters or props they can then selectively swap these elements from their cache representation to the Maya® rig representation from a single dropdown menu.

Animators can select a low-resolution rig for blocking only, or a higher resolution rig to do facial animation or an even higher resolution to animate muscle activation. Rigs resolutions are expressed as USD Variants and are loaded as Maya® references, animation curves are then loaded automatically from other Maya® files and re-connected to the rig. Rigs can also be hidden and their evaluation disabled when not needed anymore.

4 USD PYTHON COMMANDS

We soon realised that writing custom translators for every single operation we had to run in Maya® would be a bottleneck. To overcome this, we implemented a generic translator for a custom Python command system. When translating a USD prim of type ALCommand, the *AL_USDMaya* Translator would execute the corresponding Python command. Command argument values are derived by evaluating python expressions stored as string attributes on the command prim.

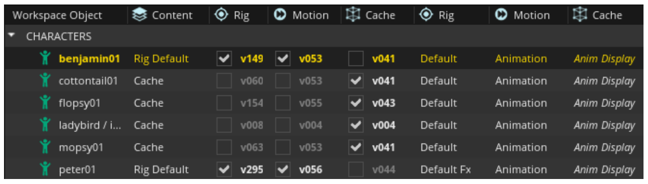
```
#python
class LogShotName(Command):
    def doIt(shotName):
        print(shotName)

#usda 1.0
def "shot_xyz"{
  def ALCommand "LogShotName"{
    string[] commandTags =
        ["translatorImport"]
    string argument:py:shotName =
        "prim.GetParent().GetName()"
  }
}
```

5 FORGE OPTIONS

As the complexity of the USD animation scene grew, we had to abstract some USD state combinations and we implemented Forge Options. These allow the bundling of prim active state and specific variants so that complex combination of USD state as well as commands could be applied by the user with a single dropdown menu.

Forge Options are implemented as a custom USD Schema typed prim and are linked to characters and props in the shot via USD relationships. Each option can configure arbitrary child prim (both active state and selected variant) as well as run additional commands. For example, the "rig_default" activates the "rig" and the "motion" prims as well as selects the "default" variant on the rig variantSet. The "wip_cache" option calls a custom command that

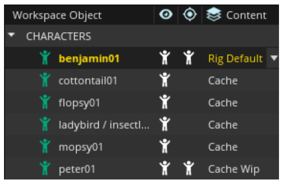


Workspace Object	Content	Rig	Motion	Cache	Rig	Motion	Cache
CHARACTERS							
benjamin01	Rig Default	✓ **v149**	✓ **v053**	☐ **v041**	Default	Animation	*Anim Display*
cottontail01	Cache	☐ v060	☐ v053	✓ **v041**	Default	Animation	*Anim Display*
flopsy01	Cache	☐ v154	☐ v055	✓ **v043**	Default	Animation	*Anim Display*
ladybird / i...	Cache	☐ v008	☐ v004	✓ **v004**	Default	Animation	*Anim Display*
mopsy01	Cache	☐ v063	☐ v053	✓ **v041**	Default	Animation	*Anim Display*
peter01	Rig Default	✓ **v295**	✓ **v056**	☐ v044	Default Fx	Animation	*Anim Display*

Figure 2: A shot in Peter Rabbit with over 6 characters where an Animator has chosen to import the rig of the 2 characters that require animating.

bakes the current deformation to a new USD layer and selects this new layer by switching a specific variant, and then hides the rig.

Data-driven configuration of the Forge User Interface allows forge options to be filtered dynamically and only shows variants that do exist on specific prims.



Workspace Object			Content
CHARACTERS			
benjamin01	●	●	Rig Default
cottontail01	●		Cache
flopsy01	●		Cache
ladybird / insectl...	●		Cache
mopsy01	●		Cache
peter01	●	●	Cache Wip

Figure 3: Same shot in the workspace view shows a simplified version of the possible combinations, where "peter01" character has been locally edited and switched to a local "wip cache" variant.

6 LIMITATIONS AND FUTURE WORK

Rig loading times are the main bottleneck during scene load times, and we are in the process of reducing this time by moving the geometry and rig bindings out of the Maya® files and into USD.

One other area we would like to improve is the overall control and visualisation of events between USD, Maya® and our bridge.

We have also started to expand the use of USD into other departments and are considering how the components presented here will apply to different workflows.

ACKNOWLEDGMENTS

We would like to acknowledge the tremendous help provided by many *Pixar* developers along the way.

We thank all the *Animal Logic* Developers (Daniel Barry, Rob Bateman, Jon-Patrick Collins, Marco Giordano, Hongbin Hu, Haoliang Jiang, Fabrice Macagno, Michael Quandt), Technical Directors, Technical Animators and Animators who all contributed to *Forge* during the *Peter Rabbit* production.