FTV500008

From In-House to Off-the-Shelf: USD in production at Animal Logic

Fabrice Macagno
Animal Logic

Learning Objectives

- Understand the history behind USD and the production challenges it solves
- Learn about the collaboration with Pixar, Animal Logic, Luma Pictures, and Blue Sky behind the USD plugin for Maya
- Understand how USD workflows have been integrated at Animal Logic into the latest version of Maya
- Get a glimpse into the future of USD workflows at Autodesk

Description

We'd like to share our journey of how we integrated Universal Scene Description (USD) in Maya into our production pipeline. We'll set the stage by describing life in Maya at Animal Logic, prior to USD, when we used a generative scene building framework called Shotsetup. This tool had its strengths and weaknesses, and led us to adopt USD for its promised functionality. We'll share a technical description of the choices we made, presenting the main artist front-facing tools fueled by AL_USDMaya (EnvironmentStudio2, Forge2). We then made the decision to open source the plugin, and will share the reasons why and the lessons learned. Finally, we will describe the collaboration with Autodesk around the unifying goal: from 3 to 1 plugin.
We also hope this can serve as a companion for our freshly released ALab USD asset: https://animallogic.com/usd-alab.

Speaker

I lead the Scene Description team at Animal Logic (AL): we love USD and all it has to offer. Our goal is to provide a fast and flexible scene building and authoring platform to artists, TDs and production staff.

fabrice.macagno@al.com.au

Introduction

Animal Logic

Animal Logic has been delivering VFX and Animation projects for 30 years, including The Lego Movie franchise, Peter Rabbit 1 & 2, Legend of the Guardian, Happy Feet.
The studio spreads across 3 locations: Sydney (HQ), L.A. and Vancouver and can peak to almost a thousand employees. AL houses Dev Ops, Production Technologies, R&D (graphics and production engineering), System Engineering, Toolchain technical departments and maintain its own 3D render engine, Glimpse.
Its production pipeline, which can handle multiple shows concurrently, has therefore weathered 3 decades of 3D content creation for large scale projects, across multiple locations and has already gone through several iterations, prior to USD's integration.

Scene description general principles at AL

From a pipeline's perspective, irrespective of the department, an artist's task always follows more or less the same steps:

1. Build a scene
2. Carry some work
3. Publish

Because of this pattern and to achieve maximum resilience, AL's philosophy is to keep its scene description (which ultimately dictates how scenes are built and introspected) "DCC-agnostic" as much as possible, and to strive to publish "deltas" whenever possible. Consequently, this means that:

- Building our scenes in Maya involves some form of *translation* from AL's breakdown
- Publishing artefacts involves some form of *extraction* of the relevant data

From Shotsetup to USD

AL's current pipeline uses USD from the ground up:

- Modelling
- Rigging
- Surfacing
- Layout
- Animation
- Lighting
- Fx

Contributions from each department are recorded and assembled using USD's powerful authoring API and flexible scene description format. The ability to capture these contributions in a non-destructive and concurrent fashion is at the heart of what constitutes modern CG/VFX pipelines.

Shotsetup: A generative framework

Our previous pipeline, which withstood 10 years of production, was no different, aiming for the same goals. It was designed around a powerful generative scene building framework: *Shotsetup* (as its name wrongly suggests, it wasn't used only for shots!).

"Generative", in the sense that our scenes' description, e.g., breakdown (i.e., which assets are used), hierarchy, was not stored physically and scenes were, instead, generated on the fly using 3 main components:

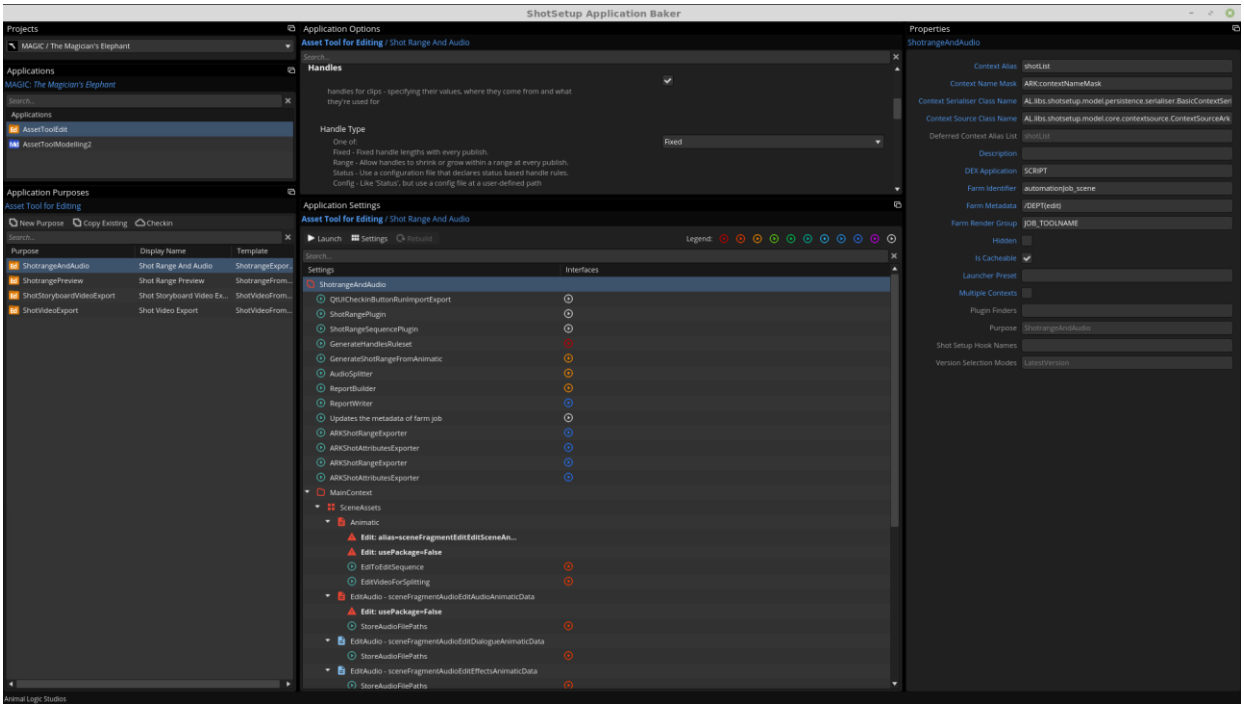
- The breakdown, pulled from our asset manager (ARK)
- Plugins, describing how to import/export data, depending on the context
- A build recipe, written in python

A scene was assembled following the recipe. Exporter plugins would then extract the work added to that scene and publish "output" assets ready to be reviewed and, once approved, handed over to downstream departments.

After 10 years, the system became the mother of all our scene builders/exporters: extremely flexible, stable and… strenuous to maintain.

Lessons learnt

- Positive feedback on flexibility and artist facing toolset
- too monolithic for agile iterations
- too complicated to configure for TDs
- too coupled to the asset manager



SHOTSETUP APPLICATION BAKER

USD integration

Late 2015, it was time to go back to the drawing board: USD wasn't yet public at that point, and because there were some doubts (like instancing performance, procedural support, render expansion), specifications for a proprietary scene description format - CSD (Common Scene Description) - were put together. Eventually, mid 2016, we decided to burry CSD and to go with USD instead, because the promises far outweighed our doubts, chiefly:

- Referencing
- Asset resolution
- Instancing
- Variations
- Overrides

At that time, project Animation 2.0, aka A20, was incubating, aiming for a transition from discontinued XSI to Maya, for performance departments (Layout & Animation). It was then decided to use A20 as a trojan horse, experimenting USD as our scene description format. With the success of A20, a global transition to using USD was put in place, following 2 phases:

- Technical migration: workflows are kept identical (as much as possible), underlying scene description is migrated to USD
- Workflow migration: new tools and workflows emerge, e.g., FilmStudio, rigged assemblies, breakdown authoring, etc.…

AN (OPTIMISTIC) INITIAL PLAN

Here is the timeline of the adoption of USD to produce some of our work:

Lego Movie 2	Late 2019	Last show fully on Shotsetup
Peter Rabbit 1	Early 2018	First show using USD (A20)
Peter Rabbit 2	Early 2020	Technical migration almost complete
Super Pets		Beginning of workflow migration
Magician's Elephant		First new workflows
Treehorn		All new workflows emerging

Maya at Animal Logic and the genesis of AL_USDMaya

Maya is our main DCC application and is used by the following departments:

- Modelling
- Rigging (which also provides a proprietary GPU based deformer custom plugin)
- Assembly
- Layout
- Animation

The pipeline around it is no exception to the principles outlined in the introduction: except for "snapshots" and "artefacts", our scenes are not saved using Maya file format, and, for instance, Maya's references are only used to bring rigs in shots (Maya scene assemblies are not used at all at AL).
Environment authoring, layout and animation are departments requiring highly optimised scenes: either because of the scale of the assets, or because of the very high playback framerate expectations.
These 2 requirements leave us with the usual challenge: finding the right balance between a lightweight but limited representation of our assets, e.g., the excellent Alembic gpu node, or a Maya native representation, which won't scale well, considering that, following AL's philosophy, we can't hinge on Maya's standard features to optimise these scenarios.

When USD was first released, it came with a maya plugin, *pxrUsdMaya* and, particularly, the *proxyShape*: a *MPxSurfaceShape* node, which can draw a USD stage directly in Maya's viewport (VP2). The general idea was there:

- USD display inside Maya without the translation cost
- The scene is kept "live", via a link to the USD stage

As the workflow supported by Pixar's proxyShape didn't match what we had in mind, i.e., "one scene - one proxyShape", we decided to write our own: that was the beginning of *AL_USDMaya.* Our goals were:

- a hybrid approach, in which parts of the hierarchy can be translated to and from Maya's data model
- an interactive way to edit USD transformations inside Maya.

This is to allow the following workflows:

- "in situ modelling", whereby, for example, artists can work at a set level and selectively "push" lower-level set pieces to Maya, edit and "pull" them back to USD
- rigs (and their motion counterparts) are represented in USD as prims with custom types, which can be translated automatically upon scene building.
- set dressing inside Maya, directly manipulating USD data model

Open-sourcing

AL_USDMaya's development started early 2016 (AL_USDMaya v0). Because AL had already open-sourced projects and because USD is open source itself, very early on, open sourcing it was on the table. The source code was eventually made available in August 2017, USDMAYA Plugin as Open Source - Animal Logic.

Aside from the obvious benefits such as knowledge sharing and external contributions, open sourcing improved drastically the quality of the code, documentation and CI/CD pipeline. On the flipside, it is fair to say that our community support fell short, mainly due to the lack of resources. Nevertheless, the experience has been incredibly rewarding and we are looking forward to open sourcing other projects.

At AL, applications and libraries are built and deployed using rez, an open-source, python, centralized package manager. Although that workflow provides many advantages, it adds a level of complexity when it comes to open-source development: 2 builds are required, an internal build and an open source one, free of any AL's specific dependencies. The CI/CD pipeline hinged on:

- *git subtree*, to integrate the source code in our rez package
- docker containers for the open-source build

On one hand, open-source builds proved to be extremely useful to reproduce errors logged on the public repository, but also were one of AL's RnD's first step into our ongoing cloud migration work.
The setup put in place for git collaboration, leveraging git subtrees, on the other hand, ended up being less than fruitful and turned out to be the cause of many headaches and confusion for developers. Not that there is anything wrong with git subtrees per se, but simply because that implied a workflow far too complicated from a developer's standpoint. It was clear that we would go about this using a different way, whenever possible.

Lessons learnt

- Improved code quality
- Faster roundtrips with external contributors thanks to the open-source build pipeline
- "git subtree" not suitable for the scenario in which the repository is not built as is
- Need to carve to some time out for community support

maya-usd

Collaboration with Autodesk

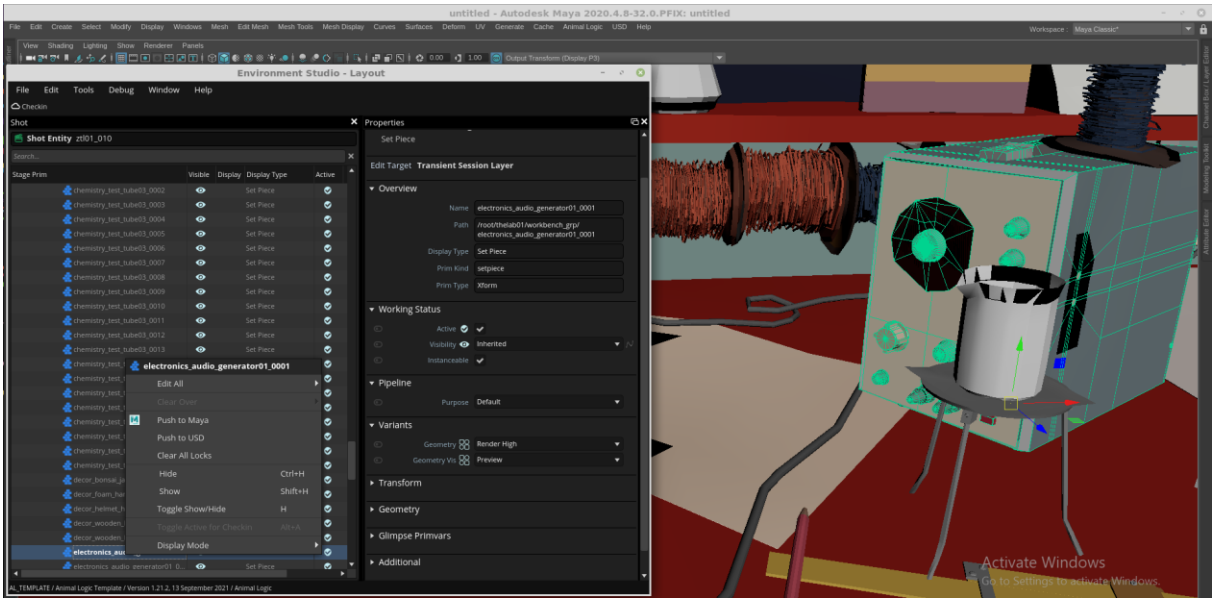
Now that AL_USDMaya was publicly released, that meant 2 USD plugins for Maya, the other one being the plugin distributed by Pixar. Despite each of them had its own strengths, it was confusing for new studios jumping on USD's wagon, as to which one to use. After some discussion with Autodesk, it was agreed in 2018 that both plugins would migrate to a new Autodesk's repository and a Technical Steering Committee (TSC) was put in place with Autodesk, BlueSky, Luma Pictures, Pixar and Animal Logic at the steering wheel. The ultimate goal was, and still is, an official USD plugin for Maya.
TSC's initial discussions were centered around the migration of the plugins, architecture, code layout and guidelines. The public repository started off with Animal and Pixar plugins sitting next to each other. Then, libmayaUsd, a shared library, was introduced: that would be the backbone of Autodesk's plugin, maya-usd. New features were added to that library and, whenever possible, existing features from either AL or Pixar plugin would be "lifted" to it. Eventually, AL and/or Pixar plugin would hinge on the new features, progressively reducing the size of non-shared code. Today, Pixar's plugin has become a very thin layer as most of its code is now found in libmayaUsd. Since then, standardized formatting and CI/CD were added, finalizing the ground work required for a fruitful collaboration. Considering the difficulty of the initial task and the place where the plugins are at today, this is fantastic achievement. Autodesk's development team demonstrated, aside from its technical knowledge, a combination of open mindedness, patience and transparency, which laid the foundation of a constructive collaboration. More recently, TSC's discussion have shifted towards less low level and more workflow-oriented questions.

Lessons learnt

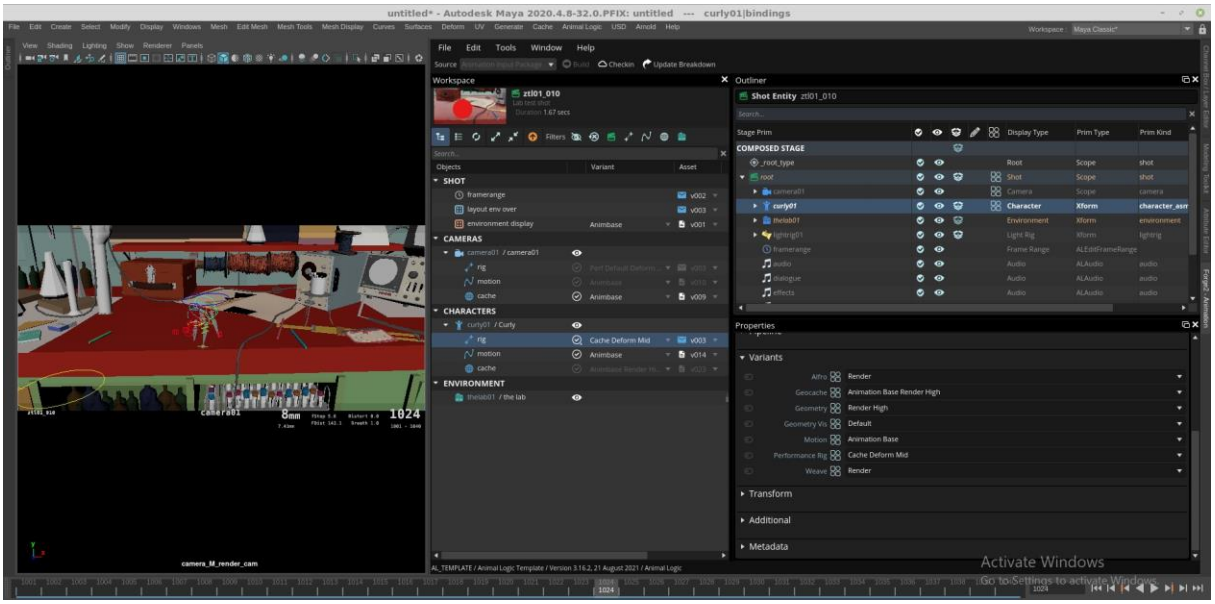
- With proper processes, mindset, diverged projects can successfully be merged back into a unified version, combining the experiences gained from various studios. This certainly represents a desirable pattern of collaboration between vendors and studios.

AL_USDMaya v1

AL_USDMaya v1 landed on March 2020, internally at Animal and in the public repository, boasting Autodesk's VP2 proxy render delegate and the first iteration on UFE selection / transformation. Although the initial results were promising for both features (faster display and native UX), it was still early days, particularly for UFE, which meant that v1 features were used for test only and disabled for production settings: workflows were still leveraging the legacy display and the custom transform nodes.
AL_USDMaya is only a component of the whole ecosystem underpinning USD's integration at AL: proprietary front facing applications, such as Environment Studio and Forge (Animation), are also pillars of that system.



ENVIRONMENT STUDIO 2


untitled* - Autodesk Maya 2020.4.8-32.0.PFIX: untitled --- curly01|bindings
File Edit Tools Window Help
Workspace
Outliner
Shot Entity zt01_010
Properties
Variants
Transform
Additional
Metadata
SHOT
CAMERAS
CHARACTERS
ENVIRONMENT
Activate Windows
Go to Settings to activate Windows.

FORGE 2

Aside from their usual scene building features, these applications are responsible for the interactive interface to the edition of USD stages from within Maya, chiefly via outliners and property editor widgets.
Prior the introduction of UFE in the production pipeline, the results of USD's integration were mixed:

- The animation workflow, in particular with the introduction of AL_USDMaya's python translators, was the most successful integration so far, albeit imperfect.
- Layout department, the most eager to adopt the new render delegate, which supports standard VP2 features such as DOF, SSAO for USD primitives, was in an "acceptable" state, still relying on the custom transform nodes
- Environment authoring was the most distressed department as the "push/pull" workflow, again build around AL_USDMaya and front facing tools was far too unstable. We had reached a point at which we contemplated taking a step back, authoring directly in Maya.

Eventually, thanks to a dedicated internal workforce, introducing a more "cross-functional" project management style, and the improvements made to UFE by Autodesk, AL_USDMaya v1 was adopted in production, restoring a more stable working toolset for environments artists.

Lessons learnt

- A more cross-functional approach, i.e., tight connection between R&D and production participants, to change management and strategic decisions appears to be a much better solution to mitigate the risks of integrating emerging technologies.

Towards maya-usd

AL_USDMaya is now at its v2 iteration: it doesn't differ drastically from v1, except that AL's legacy code dealing with USD transformation has been deprecated, thanks to the new transformation stack, result of an initiative led by the TSC. The VP2 proxy render delegate and UFE are used in production on all shows.
Ultimately, we will deprecate AL_USDMaya and switch to maya-usd as soon as it has reached feature parity with our plugin. The transition will take place in 2 phases

1. Static imports/exports will be switched over to maya-usd
2. ProxyShape based workflows replacement