Editorial Pipeline Conversion: Animal Logic's Transition to OpenTimelineIO

Tim Lehr
Animal Logic
Vancouver, BC, Canada
tim.lehr@animallogic.ca

Barish Balachandran
Animal Logic
Sydney, NSW, Australia
barishb@al.com.au

Oliver Dunn
Animal Logic
Vancouver, BC, Canada
oliver.dunn@animallogic.ca

Nathan Lacey
Animal Logic
Sydney, NSW, Australia
nathan.lacey@al.com.au

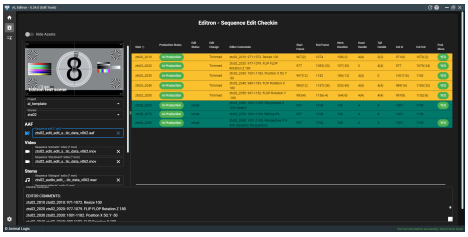

Figure 1: Editron user interface. Built using Electron.js and OpenTimelineIO.

ABSTRACT

We introduce Animal Logic's editorial pipeline refactor from a rigid and overly complex in-house solution, towards a more modern, flexible approach, based on the open source technologies *OpenTimelineIO* and *Electron.js*. This upgraded design greatly increases flexibility over the previous effort, enabling cross-platform user adoption and further decoupling our tools from the editorial software of choice. The new pipeline is now rolled out onto our most recent productions and we are already starting to see the benefits of its extensibility and ease of troubleshooting.

CCS CONCEPTS

• **Computing methodologies** → **Computer graphics**; **Editorial**.

KEYWORDS

Editorial OpenTimelineIO Electron Pipeline

ACM Reference Format:
Tim Lehr, Barish Balachandran, Oliver Dunn, and Nathan Lacey. 2022. Editorial Pipeline Conversion: Animal Logic's Transition to OpenTimelineIO. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '22 Talks), August 07-11, 2022.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3532836.3536278

1 INTRODUCTION

While Animal Logic's pipeline has gone through several major refactors in recent years, namely its conversion over to a fully Pixar™ *Universal Scene Description* (USD) based solution, due to several limiting factors, the editorial pipeline had not. With the advancements of *OpenTimelineIO* (OTIO, https://github.com/PixarAnimationStudios/OpenTimelineIO), Animal Logic's growth, and with it a pressing need to upgrade our code base to Python 3, it was time for a redesign of our editorial tools.

There were two main requirements to be considered with this refactor: The user interface for editorial needed to be cross-platform; developers and TDs would be using the tools on Linux, while the editorial department would be running either Windows or Mac. The second main requirement was to reduce the large amount of in-house custom Python code and remove dependencies on any particular editorial file format, allowing us to greatly improve our agility and flexibility in this area going forward.

2 USER INTERFACE

After investigating multiple frameworks and deployment technologies, the decision made was to develop a native application. A modern web frontend framework (Vue.js) in combination with Electron.js was chosen for this task, giving us the ability to develop the application once and deploy it across multiple platforms. *Editron* (frontend application) takes media files from the editorial application, converts the cut data to OTIO and compares it against previous versions. Finally Editron packages the files and sends them to the backend for publishing. Shown in Figure1.

2.1 Evaluation Options

There were several things to consider for the user interface. Web vs native, in this instance native was chosen to allow for speed of processing the large files the users would be uploading. Web applications lack the native file-system support that was required here. Electron vs. Qt/PySide-2, in this instance Electron was chosen for its portability to other platforms. While Qt is known territory for developers at Animal Logic, internal build systems were not designed for cross platform and upgrading to Electron was the preferred path.

SIGGRAPH '22 Talks, August 07-11, 2022, Vancouver, BC, Canada

ACM ISBN 978-1-4503-9371-3/22/08.
https://doi.org/10.1145/3532836.3536278

3 BACKEND

While Editron is responsible for the initial conversion and asset bundling, the rest of the pipeline is executed on a Linux server running our proprietary task graph engine. The backend task graph includes the generation of multiple media proxies from the published asset bundle, as well as the creation of a set of EDL- and USD-files. An overview and the detailed node network can be seen in Figure2.

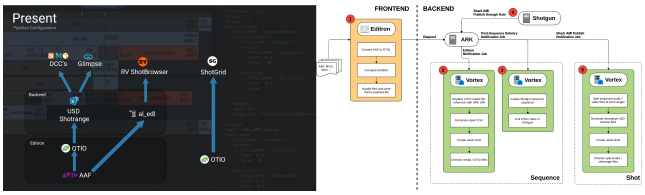

Figure 2: High-level pipeline diagram and backend task node graph.

3.1 OpenTimelineIO Contributions and Enhancements

We needed to make a few additions to the OTIO AAF adapter (Avid's Advanced Authoring Format), primarily enhancing support for editorial markers, which we use internally to highlight tracks of interest and attach metadata to media clips. These changes have since been merged back into the public OTIO repository and we are continuing our commitment to contributing back to the community going forward.

The introduction of frame handle schemas to accommodate the varying requirements between different departments, was the most significant internal addition to the OTIO ecosystem. We currently support multiple frame handle sizes, implemented as specializations of a new common rule schema, with departments sharing access to a pool of configured rules for a given show.

3.2 USD Integration

With Animal Logic's overall pipeline being built around USD, it was imperative that this new refactor integrated seamlessly with our current suite of client applications. In particular Animal Logic's shot building application *Forge* [Baillet et al. 2018], reads shot range information directly from USD files. At publish time, a pipeline step reads the sequence OTIO file and writes out our custom frame range prim to the scene description per shot. This prim is then read by our tools as well as by *Glimpse* [Heckenberg et al. 2017] at render time to denote the range of the rendered images.

3.3 Asset Resolution

In order to be able to access published media assets across all of our studio locations, we require all media references to conform to our custom URI schema. This translation is made by our OTIO media linker plugin during the initial Editron conversion from AAF to OTIO. On subsequent access to the OTIO data, our internal asset resolver web service translates any URI's into localized file paths. We take advantage of the hook API provided by OTIO to send requests to our resolver service making it consumable by any client at the requested studio location.

4 RESULTS AND FUTURE DIRECTION

The new refactored pipeline is now rolled out into our most recent projects. It has allowed them to fully embrace python 3 and allowed for TDs to easily and quickly configure the Editorial process for each of these shows. It has also greatly improved troubleshooting this area of the pipeline as issues arise during the heat of production. Our previous codebase in this area comprised over 100,000 lines of code, the main factor making it so hard to work with. Now thanks to integrating OTIO and Electron, our internal code is down to around 5,000 lines, making it much easier to work with and maintain.

While OTIO is now integrated into Animal Logics pipeline, there is still work to be done to further improve and streamline our editorial tools. There are two main aspects we are looking to work on: First, the deprecation of our USD shot range asset and our own in-house EDL format. These assets add extra layers of complexity with little to no benefit. Future steps will be to retire these file formats and have all in-house tools and third party tools read OTIO instead.

We are also continuing to follow the progress made by Autodesk in integrating OTIO into Shotgrid™, with the potential of presenting rich timing information read directly from OTIO. Our future pipeline design can be seen at a high level in Figure3.

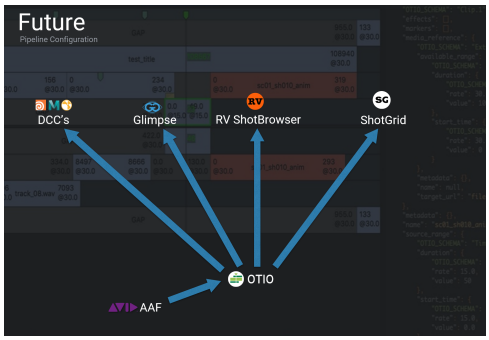

Figure 3: Pipeline diagram depicting future directions.

ACKNOWLEDGMENTS

We would like to thank Fabrice Macagno for his mentorship, the Animal Logic RnD Pipeline and Production Systems teams for their help on this project, as well as the whole OTIO community for being so welcoming and providing great technical guidance as we worked on this project.

REFERENCES

Aloys Baillet, Eoin Murphy, Oliver Dunn, and Miguel Gao. 2018. Forging a New Animation Pipeline with USD. In *ACM SIGGRAPH 2018 Talks* (Vancouver, British Columbia, Canada) *(SIGGRAPH '18)*. Association for Computing Machinery, New York, NY, USA, Article 54, 2 pages. https://doi.org/10.1145/3214745.3214779

Daniel Heckenberg, Luke Emrose, Matthew Reid, Michael Balzer, Antoine Roille, and Max Liani. 2017. Rendering the Darkness: Glimpse on the LEGO Batman Movie. In *ACM SIGGRAPH 2017 Talks* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 8, 2 pages. https://doi.org/10.1145/3084363.3085090