Developing a Curve Rigging Toolset: a Case Study in Adapting to Production Changes

Thomas Stevenson
Animal Logic
Wellington, New Zealand
thomas.stevenson@al.com.au

Valerie Bernard
Animal Logic
Vancouver, BC, Canada
valerie.bernard@animallogic.ca

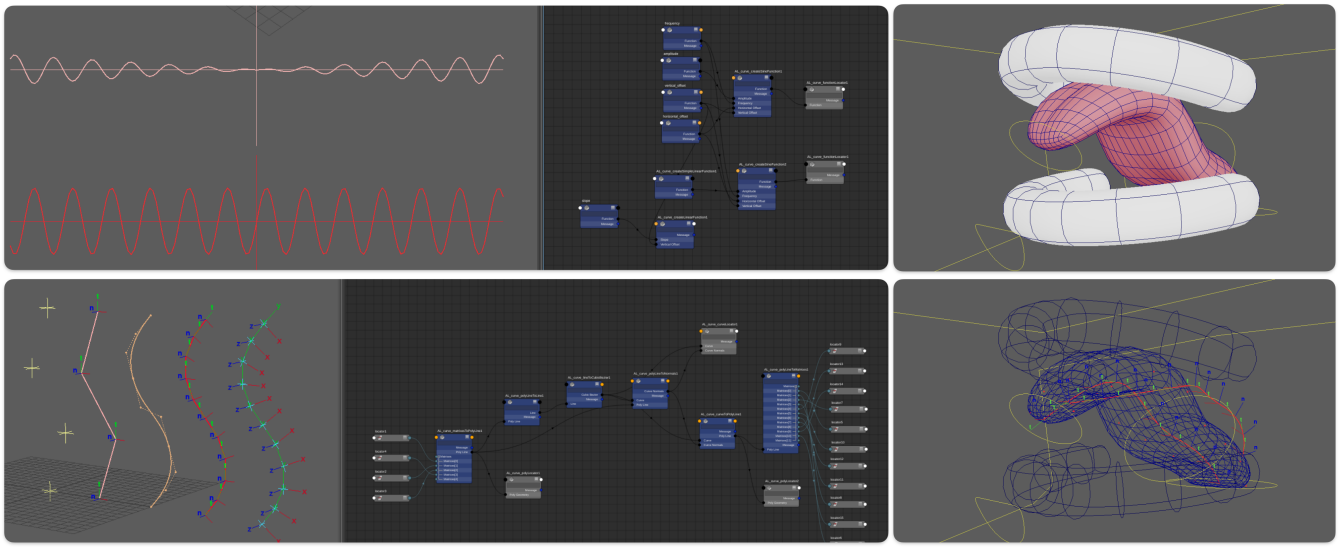


Figure 1: Examples of curve and function rigs (left) and a production tongue rig built on the same technology (right).

ABSTRACT

We present an overview of Animal Logic's curve rigging toolset and its development process, serving as a case study to discuss challenges specific to doing software development for animated feature film production. We will show how R&D projects at Animal Logic lean on agile software practices to enable ambitious development projects, with flexible plans that adapt to the reality of working with creative stakeholders. We will highlight the importance of production engagement, reflect on our technical decisions made over a year of active development while reacting to drastic production schedule changes, and share lessons learned along the way.

CCS CONCEPTS

• **Computing methodologies** → **Animation**; *Parametric curve and surface models.*

KEYWORDS

Curve, Spline, Rigging, Animation, Project Management

ACM Reference Format:
Thomas Stevenson and Valerie Bernard. 2024. Developing a Curve Rigging Toolset: a Case Study in Adapting to Production Changes. In *Proceedings of Digital Production Symposium 2024 (DigiPro '24).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

DigiPro '24, July 27, 2024, Denver, CO

ACM ISBN XXXX-XXXX-X/24/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Since 2016, Animal Logic has been crafting a new animation pipeline based on Autodesk Maya® and Pixar USD®. The Animation and Rigging R&D team is in charge of maintaining rigging tools, like our rigging framework and proprietary deformation system *Bond* [Baillet et al. 2020] and animation tools, such as our shot sculpting toolset [Bernard et al. 2022]. Our animated features can vary greatly in animation styles, driving the need for solid generic rigging toolsets that can cover a vast array of creative requirements. With our experience building new workflows and tools for both rigging and animation over the past few years for our new pipeline, we look to share the lessons we have learned on how to adapt development plans to the reality of working with creative stakeholders, and how we react to drastic changes in production schedules and priorities. We will use the development of our new curve rigging toolset as a case study.

1.1 The original request from production

Since we transitioned to our Maya pipeline, curve rigging technology at Animal has been limited to proprietary nodes, derived from Catmull-Rom splines, for creating and sampling curves, along with making use of Blur Studio's [TwistSpline 2023] nodes. These nodes were used on our latest shows, *The Magician's Elephant* and *Leo* (see Fig 2).

At the start of 2023, with an animated feature starting mid-year that we knew included a character with a long highly articulated tail, the rigging team asked for a variable FK node (see [Brosky 2013]) to simplify their existing trunk and tail setups.

Figure 2: Characters built using existing curve technology ©Netflix.

1.2 The motivation for a curve rigging toolset

With asset production starting in the fall of 2023, we saw an opportunity for addressing that variable FK request along with a vast array of other features. Our existing solutions had limitations, like lack of control over tangency and normal interpolation. These ad-hoc solutions can be quick to deliver and perform their purpose but don't lend themselves to feature growth.

With a powerful library of generic curve nodes, we would be able to address any new curve feature request very quickly, so we can easily reproduce desired industry solutions for tentacles [Hessler and Talbot 2016], ropes [Sheerin et al. 2022], motion paths [Lin et al. 2023] and anything else production might require. Creating a variable FK setup for instance would become trivial with the right building blocks.

We put together a project pitch for production approval, documenting limitations of existing solutions and our proposed solution of a framework for curve rigging as in Fig 3.

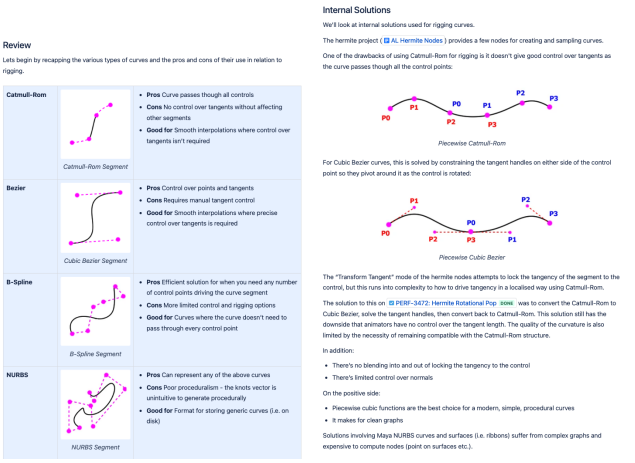


Figure 3: Excerpts from our project proposal.

1.3 Managing the risks

Before greenlighting the project, we assessed the risks involved and how to mitigate them. We illustrate the following steps we took to address our main concerns.

1.3.1 Defining a Minimum Viable Product (MVP). As we created a development plan during pre-production, our knowledge of creative requirements was limited. Characters can be cut, the animation style can be updated, leading to different technical requirements in the rigs. We ran the risk of implementing features that end up not being used in production. The more specific the feature, the higher the risk. We minimized this risk by defining an MVP to be delivered for testing a couple months before asset production started in the fall. Once we knew more about the show's creative requirements we could schedule development of advanced features.

1.3.2 The "bus factor". Our team is cross-functional, with technical expertise to cover all aspects of doing R&D for animation and rigging but not all engineers have the expertise to maintain all projects. What happens if we lose our main developer? We ran regular technical workshops to ensure more engineers could contribute to code reviews and overall code maintenance, providing opportunities for up-skilling team members.

1.3.3 Technical stakeholders. Rigging being a technical discipline, we ran the risk of seeing riggers develop their own plugins to address a short term need, instead of waiting for a more elaborate solution that covers all use cases. This is not a concern we have when doing projects for animators and we must adapt our approaches for each project based on our stakeholders.

1.3.4 Production engagement. A big concern with long development cycles is having enough engagement from artists to guarantee a successful outcome in production. Past experiences have taught us that delivering major developments into production can be painful for artists if they did not buy into the project from the start. Sprint reviews and weekly catch ups with rigging supervisors and leads keep us in constant communication so we can revise priorities

based on production needs. With excited riggers supporting our plan, development started in June 2023.

2 PROJECT OVERVIEW

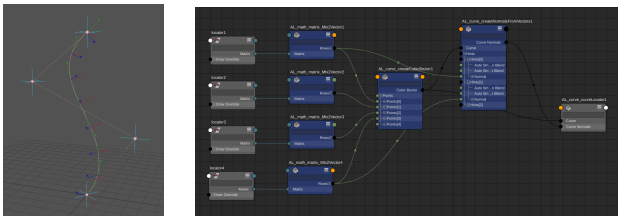
2.1 Curve primitives

With the various curve rigging methods in use in production, we opted to implement multiple curve types and support various approaches to rigging curve normals. This gave artists the flexibility to use the toolset with their preferred approach.

We opted not to support Catmull-Rom splines. The inability to control tangency is a significant downside and we preferred to provide "auto-tangent" behaviour as part of the Cubic Bezier infrastructure.

We also decided to skip NURBS splines, as in Maya's native curve implementation. In practice, rigging only uses a subset of NURBS, the closed uniform B-Spline. This has the desirable properties where each control point has a consistent influence over the curve and where the curve passes through the first and last control points. As such, we provided a closed uniform B-Spline type that does not require artists to provide a knot vector as part of the tooling.

In part due to our choice to support multiple curve types, we opted to implement curve normals as their own primitive, as in Fig 4. This has the effect of not producing a combinatorial explosion of types (i.e. supporting oriented and non-oriented versions of each curve type). While this results in additional connections in the node graph, artists aren't forced to use a specific curve normal interpolation method.



(a) Curve. (b) Node graph.

Figure 4: Separate curve and curve normals primitives.

2.2 MVP features

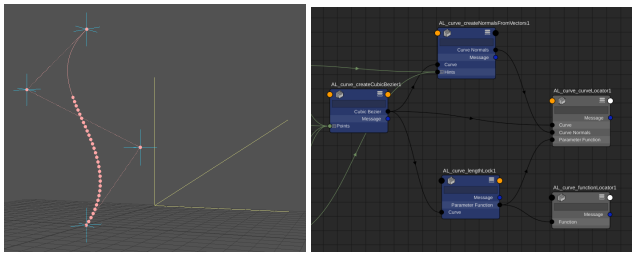
We included the basic primitives, curves and curve normals, and the basic nodes for visualising, creating, querying and sampling in our MVP. We had also planned on delivering a Bond curve deformer to ensure we addressed all basic needs from control rig to deformation stack, along with all necessary nodes to implement the variable FK setup.

However, before we could tackle the curve deformer, riggers asked for new features with high priority based on how asset work was progressing in pre-production and their testing of the MVP. This pushed the deformer back, and the variable FK was now set to lowest priority, beyond more advanced features. In particular, riggers asked for sine curves to address a very specific creative requirement in a character on the show.

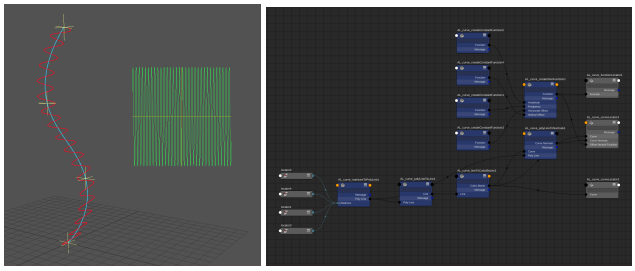
2.3 Advanced features

With the core functionality in place, we focused on the ability to lock curve samples to a specified length, and the node infrastructure required to sufficiently replace the existing Catmull-Rom node behaviour using the Cubic Bezier curve type.

Length locking can be achieved by remapping the parameter space of the curve, as opposed to splitting or rebuilding the curve, and can thus be conceptualised as a function mapping $t \in [0, 1]$ to $t \in [n, m]$ as seen in Fig 5.



(a) Length locked curve (left) and its node graph (right).



(b) Sine wave applied to a curve (left) and its node graph (right).

Figure 5: Examples of functions in use.

Sine curves generally provide for offsetting a parametric curve on its normal and binormal, using a sine curve with control over the frequency and amplitude in both axis.

Furthermore, the original request for Variable FK, lower in priority but still required down the line, can be reformulated as a sum of functions operating in rotation space and can therefore also be represented as a function.

This provided us with three use cases for implementing a more generalised interface for functions. As with curve normals, we implemented these functions as a separate primitive which can be reused in all cases.

Length locking was delivered as a single `lengthLock` node and the sine curve functionality as a single `sineOffset` node. For practical usage, it was also necessary to have tip and tail falloffs and variation in amplitude and frequency.

These function primitives, while requiring additional development around the node infrastructure, allowed flexibility in defining functions for remapping the parameter space and for offsetting in curve space.

We then looked to extend the Cubic Bezier node infrastructure to emulate the existing auto-tangent of the Catmull-Rom nodes. This was accomplished by working out ideal Cubic Bezier tangent

placements and then providing the ability to blend between the manual and auto tangents.

2.4 Changes in slate of shows

The fall of 2023 came with the announcement of changes to our slate of shows, which impacted the projects our teams had in development. This meant that our production teams had a few months to focus on internal creative projects before work began on the new upcoming shows starting mid-2024. We then looked to revisit our development priorities to align with the requirements of these new projects.

2.5 New rig component priorities

At that time, the rigging team started a complete overhaul of our facility components, which drastically changed their curve requirements. Without enough time for us to implement the requested features, riggers integrated custom plugins they wrote into their components. We then followed along their work to ensure we built feature parity over the coming weeks into the toolset, to later easily replace those temporary plugins with the official nodes.

As part of the facial component development, Cubic Bezier curves had started to be used. This highlighted some issues with the way we had implemented the Cubic Bezier node infrastructure, and some time was spent reworking it.

As riggers were focusing on components, we pushed the curve deformer in favour of investigating a further reported issue with our approach to arc-length parameterisation.

2.6 An opportunity for research

For Bezier curves, existing arc-length parameterisation solutions have degenerate cases which were picked up by artists during testing. There are workarounds that we likely would have used in the middle of production, however these are not ideal for a general purpose toolset and we could spare a couple of months to solve the problem properly.

For B-Splines, being a popular choice among experienced Maya riggers, it was noted that Maya's default chord parameterisation method was preferable in certain cases, having the advantage of preserving the localisation of a control's influence.

We investigated to find an appropriately efficient and accurate method for solving both parameterisation problems and settled on an approximation method that satisfied our requirements, working for both Bezier and B-Spline curves and providing the span parameterisation in parity with Maya's chord parameterisation.

2.7 New show priorities

With new animated features starting up in the spring of 2024, we once again refocused on production requirements, although different from what we initially started with. For instance, we were now looking at potentially elaborate rigs for characters wearing dresses.

We added support for closed curves and localised parameterisation, as in Fig 6, and extended the curves into two dimensions in order to support rigging surfaces.

We must note again that the curve deformer work was pushed in order to implement these features. The variable FK request was all but forgotten at this point.

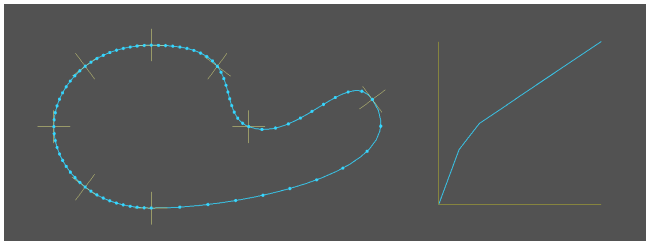


Figure 6: Example of a closed Cubic Bezier curve with parameterisation localised to the curve segments.

3 LESSONS LEARNED

3.1 Production engagement is critical

Maintaining good production engagement over the lifetime of a long development project is critical for its success.

Sprint reviews provide a platform for continuous feedback, allowing for changes to the development plan based on direct input from artists. Keeping informed on the work being done by rigging artists can also shift requirements and priorities.

Adapting the development schedule to accommodate day to day requests helps to keep engagement high, showing a commitment to supporting the artists' immediate needs and creating a sense of collaboration.

3.2 Sometimes you need to push back

While responsiveness to artist requests is important, it's also sometimes necessary to push back and enforce set priorities for the sake of code quality.

It took us a year to finally prioritize the implementation of our curve deformer, which was meant to be an MVP feature. This was fine in terms of priorities for artists, but implementing the deformer would have let us put in place some code infrastructure that would have greatly simplified the implementation of the sine curve nodes. As it stands, rushing the sine curves led to the introduction of some tech debt early into the project life cycle. Looking back, we had time to do the deformer before the sine curves needed to be delivered so this could have been avoided.

We've since learned to spend time with rigging supervisors looking at delivery schedules and how we can best accommodate major development requests without sacrificing code quality when possible.

3.3 You can't predict everything artists will need

Animation production is inherently as agile as development. Our creative partners are constantly iterating and responding to director and test audience feedback, having to adjust their own priorities and goals. Designing software that can be responsive to these changes is key for pain-free production adoption.

Preparing a thorough project proposal with short and long term goals that was validated by production helped us come up with a simple software design that was highly flexible to priority changes.

4 CONCLUSION AND FUTURE WORK

With careful planning and constant communication with the rigging team, we were able to successfully deploy our new curve nodes into production rigs. The response from riggers and animators using the rigs has been highly enthusiastic and our toolset is now maintained by multiple engineers focusing on upcoming show priorities.

Further development projects are also being launched using these curve nodes as a framework, such as screen space tools for silhouette sculpting in animation, rigging for surfaces, 2D texture animation, linework. We'll also probably implement that variable FK node at some point...

ACKNOWLEDGMENTS

We would like to thank our team members Antoine Domon, Daniel Springall, James Dunlop and Miguel Gao for their contributions to the project, and Enrique Caballero and the whole rigging team for their continued trust and collaboration.

REFERENCES

Aloys Baillet, Josh Murtack, Hongbin Hu, Haoliang Jiang, and Michael Quandt. 2020. Bond: USD-Integrated Hybrid CPU, GPU Deformation System. In *ACM SIGGRAPH 2020 Talks* (Virtual Event, USA) *(SIGGRAPH '20)*. Association for Computing Machinery, New York, NY, USA, Article 34, 2 pages. https://doi.org/10.1145/3388767.3407324

Valerie Bernard, Miguel Gao, Daniel Springall, and David Ward. 2022. Powering up Rig Deformation: Shot Sculpting on DC League of Super-Pets. In *ACM SIGGRAPH 2022 Talks* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 45, 2 pages. https://doi.org/10.1145/3532836.3536247

Jeff Brosky. 2013. Trunk rig - Variable FK - How it works. Video. Retrieved May 10, 2024 from https://vimeo.com/72424469

Mark Hessler and Jeremie Talbot. 2016. AutoSpline: animation controls only when and where you need them. In *ACM SIGGRAPH 2016 Talks* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 7, 2 pages. https://doi.org/10.1145/2897839.2927439

Andy Lin, Hannah Swan, Justin Walker, Cathy Lam, and Ricky Arietta. 2023. Swoop: Animating Characters Along a Path. In *ACM SIGGRAPH 2023 Talks* (Los Angeles, California) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 45, 2 pages. https://doi.org/10.1145/3587421.3595425

Daniel Sheerin, Joshua Beveridge, Enoch Ihde, Stirling Duguid, Brian Casper, Andrea Parkhill, Ed Lee, and Carlos Fraiha. 2022. Rigging the Rigging: An Animation Friendly Rope System for The Sea Beast. In *The Digital Production Symposium* (Vancouver, BC, Canada) *(DigiPro '22)*. Association for Computing Machinery, New York, NY, USA, Article 3, 7 pages. https://doi.org/10.1145/3543664.3543680

TwistSpline. 2023. *blurstudio/TwistSpline*. Blur Studio. Retrieved May 10, 2024 from https://github.com/blurstudio/TwistSpline