Cartoons in the Cloud

Yun Lien
Pixar Animation Studio
Emeryville, CA, USA
yun@pixar.com

Michael O'Brien
Pixar Animation Studio
Emeryville, CA, USA
mobrien@pixar.com

Laura Savidge
Pixar Animation Studio
Emeryville, CA, USA
lsavidge@pixar.com

ABSTRACT

The SparkShorts program at Pixar Animation Studios allows for directors to try new and different looks. Our short, *Twenty Something* is a 2d, hand-drawn animated film. When the pandemic forced all of our artist to work from home, we scrambled to create a workflow for managing, sharing, and reviewing 2d assets. While we have a long history of collaborating on 3d films, we did not have a solution for 2d imagery.

We created a cloud-based pipeline based around on-premises bucket storage, microservices, and event-driven workflows. The result was Toontown, a suite of technologies that allowed our artists to complete *Twenty Something* working from home.

ACM Reference Format:
Yun Lien, Michael O'Brien, and Laura Savidge. 2021. Cartoons in the Cloud. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '21 Talks), August 09-13, 2021.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3450623.3464680

Figure 1: *Twenty Something* explores why adulting is so hard. ©Disney/Pixar

1 INTRODUCTION

In March of 2020, the state of California issued a Shelter-In-Place (SIP) order for the entire state, sending our artist home, many carrying their workstations with them. At the time, our short flim, *Twenty Something* was about to begin production. Before SIP, we had done some early exploration of the department structure we wanted for the project.

SIGGRAPH '21 Talks, August 09-13, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8373-8/21/08.
https://doi.org/10.1145/3450623.3464680

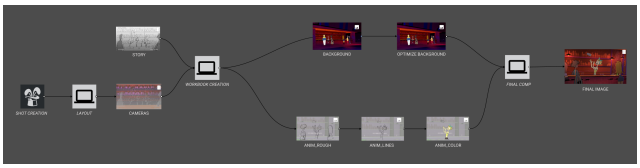

Figure 2: *Twenty Something* departments. ©Disney/Pixar

In order to complete the show, we created a Pipeline as a Service (PiaaS) for managing and sharing 2d assets, called Toontown. On the artist end, we wanted to use Adobe's PhotoShop, Adobe's After-Effects, TV Paint's TV Paint Animation, and Autodesk's Maya®all on OSX. We had a little experience with those tools, but nothing formal to manage how to pass data between them. Our resulting design needed to allow each department to operate within their application, but still share data downstream. In order to do this, Toontown supports complex associations of assets, defines a suite of services that can be provisioned per show, and provides a user interface to manage those services.

2 TOONTOWN OVERVIEW

Looking at the pipeline from a file-centric view, Toontown consumes data from editorial, provides a working space for our production departments, and then synchronizes back with editorial once final images are available.

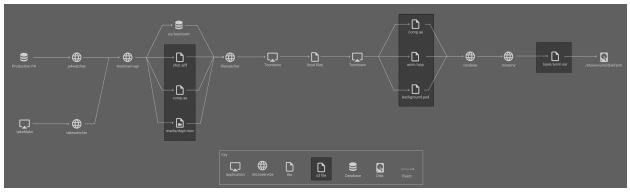

Figure 3: Toontown pipeline design. ©Pixar

Reading from left-to-right, editorial produces an audio file representing the shot's dialog which is checked into Perforce and copied to the diskfarm file system by our video encoder system, takeMake. Those files are processed by a microservice and transferred up to bucket storage and indexed into an ElasticSearch database. Via another microservice, files are mapped from the user's machine for consumption by Toontown and the Digital Content Creation (DCC) application. Toontown publishes files back to bucket storage for consumption by a microservice to manage translation between application file types and final frame creation. The resultant files become usable by both editorial and downstream departments.

3 ASSET MANAGEMENT

Assets are separated into a source file for the DCC and the generative files needed for the next stage of production. For example, an animator needs to author a TV Paint file (*.tvpp), but that file is not consumable by AfterEffects. The tvpp file needs to be translated into a set of layered exrs corresponding to each layer in the tvpp file.

Each source file is tracked using an on-prem object store that adheres to the s3 protocol. The contents of the bucket are indexed as a set of metadata documents in an ElasticSearch database. Where our traditional pipeline references assets via unix-style paths, Toontown provides a many-to-many document index that a client queries. The resulting document provides the bucket location and metadata. The client can then decide how to map the resulting bucket to the user's machine.

Our traditional pipeline looks in multiple paths to resolve the final version of that asset, using path fragments as metadata. This can be costly and brittle. In Toontown, the location of the asset data is fully independent from how it is resolved. The ElasticSearch index is used as a flexible backend to find assets. The database can contain multiple records for a single bucket which allows for multiple methodologies for clients to find a bucket. This allows a single background image to be referenced into multiple shots, "Find me the background painting for this shot." The file can be found through larger queries, "Find me all of the files for this shot." It also allows us to create indices for specific use cases. For example, we can create a set of documents based on abstract concepts, "Find me all of the paintings in a kitchen." The file's contents are fully separated from the resolution.

The client can then map the bucket to different locations on the user's local machine based on the use case. Our artists doing background paintings would like all of the paintings to be in one directory named after the shot. Our compositors would like the background painting to be mapped into multiple, per-shot directories. Since the bucket's data is shared, publishing a new version of a bucket notifies all clients to refresh their version of the data.

4 SERVICE SUITES

Toontown is powered by independent microservices connected via events. The design is constructed around a microservice providing a small, department-specific piece of functionality.

Each microservice is written in JavaScript and provisioned on our in-house Kubernetes cluster as a web-service. Starting a new show requires spinning up each service needed for the show's pipeline. Using a PaaS for provisioning significantly reduces the time to set up a new show. Additionally, since each show uses its own instances of the services, the services can be turned off after the show is complete. Toontown's modular architecture makes it easy for a show to experiment with new workflows or applications. For example, if a show decides to try another animation package, only the animation services need to be replaced to recognize the new DCC file. If a show decides they want to use just AfterEffects to construct graphics, the microservices for handling other file types do not need to be provisioned at all.

Our events are hosted on our on-prem Kafka servers. Kafka provides event producer and consumer modules for any language we use at Pixar, and many of our tools emit events for key state changes. Since Toontown is also using the same event bus, we can share and merge workflows written in different departments. For example, a take from editorial can trigger a take event causing Toontown to update the latest take. Toontown, itself, can also emit the same event when animation updates their latest animation thereby creating a new take. The same event feeds into the system.

While we chose to use on-prem infrastructure for this first show, our use of standard components offered by cloud service providers allows us flexibility to run portions of the Toontown pipeline on a commercial cloud platform with minimal changes.

5 USER INTERFACE

Our 2d shorts are staffed by our more traditional artists and animators. Toontown needs to provide a simple workflow for them. Since Toontown was developed, frantically, during the start of the pandemic, it was important the interface could be updated quickly, deployed easily, and share as much code between the different department workflows.

Toontown is built as an embeddable webapp written in JavaScript, like the microservices, allowing code sharing between the interface and the backend code.

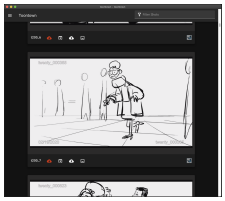

Figure 4: Toontown user interface. ©Disney/Pixar

As a standalone application, Toontown provides an Electron app running a small webserver in the background. The artist is presented with a set of cards indicating active inventory based on data from our Oracle production database. Behind the scenes, Toontown is actively aggregating and synthesizing data from our production database, the ElasticSearch database for bucket information, and the bucket storage to map files to the artist's machine.

As a webapp, Toontown can also be embedded in Adobe's products using their Common Extensibility Platform (CEP). This allows Toontown to interact with all of the same backend services as the electron version, but also control DCC specific properties, like synchronizing frame ranges between editorial and AfterEffects.

6 CONCLUSION

Toontown was instrumental in completing *Twenty Something* during the pandemic. It also provided the opportunity to explore how cloud-based pipelines can shape our view of how to find and manage data. Looking forward, we anticipate taking the concepts of what was learned on Toontown and exploring how we can utilize them in our 3d pipeline.