

5-2019

Analyzing and Developing Aspects of the Artist Pipeline for Clemson University Art

Kunta Kwaku Lowe

Clemson University, kuntakwaku055@gmail.com

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses

Recommended Citation

Lowe, Kunta Kwaku, "Analyzing and Developing Aspects of the Artist Pipeline for Clemson University Art" (2019). *All Theses*. 3098.
https://tigerprints.clemson.edu/all_theses/3098

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

ANALYZING AND DEVELOPING ASPECTS OF THE ARTIST PIPELINE FOR CLEMSON UNIVERSITY DIGITAL PRODUCTION ARTS

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Fine Arts
Digital Production Arts

by
Kunta Lowe
May 2019

Accepted by:
Dr. Eric Patterson, Committee Chair
Dr. Victor Zordan
Dr. Jerry Tessendorf

Abstract

Major digital production facilities such as Sony Pictures Imageworks, Pixar Animation studio, Walt Disney Animation Studio, and Epic Games use a production system called a pipeline. The term “pipeline” refers to the structure and process of data flow between the various phases of production from story to final edit. This paper examines current production pipeline practices in the Digital Production Arts program at Clemson University and proposes updates and modifications to workflow. Additionally this thesis suggests tools that are intended to improve the pipeline with artist friendly interfaces and customizable integration between software and remote-production capabilities.

Acknowledgments

The support, feedback, and guidance of my advisor, Dr. Eric Patterson, made this thesis research a success and thus has my sincere gratitude. His efforts not only enabled me to face and succeed in the challenges of my graduate studies, but also helped in getting me prepared for a career in the industry.

A special thanks to Dr. Victor Zordan and Dr. Jerry Tessendorf for advising and guiding me through the thesis research. Their insight and knowledge greatly aided in its success.

To my fellow colleagues, Daniel Hale, DN Cherry, Derek Andrews, Kira Foglesong, Caroline Requierme, John Welter, Margaret Wages, Amanda Smoak, Chance Cochran, and students in the Digital Production Arts program, thank you for your support, insight and encouragement during the research process.

Finally, to EdVania Powell for being a pillar and mentor throughout my time as a graduate student, Jeff Denton for his support and wisdom during my research, and Wes Holladay for his endless encouragement. Your advice and goodwill is greatly appreciated.

Table of Contents

Title Page	i
Abstract	ii
Acknowledgments	iii
List of Figures	vi
1 Introduction	1
1.1 Asset Management	2
1.2 Project Management	3
1.3 Art and Pipeline	4
2 Background	6
2.1 Industry Production Pipelines	6
2.2 DPA Pipeline	8
2.3 Shotgun Toolkit	10
3 Case Studies	11
3.1 “The Boy Who Cried”	11
3.2 “Brave Player”	15
3.3 “Disconnect”	25
3.4 “For A Rainy Day”	29
3.5 Statistics	34
4 Design	41
4.1 Workflow	41
4.2 DPA-pipe Proposal	46
4.3 Toolkit Proposal	48
5 Implementation	50
5.1 DPA-pipe Shotgun Publisher	50
5.2 Tractor Queue Submit	52
5.3 Toolkit: Ubuntu Linux	54
6 Conclusion	55
6.1 Analysis	55
6.2 Future Implementations	57
Appendices	58
A Statistics Questionnaire	59
B Palmetto Singularity Recipe	62

Bibliography 68

List of Figures

1.1	Pixar Traditional Pipeline	2
1.2	Maya Shader Dependency Graph	3
1.3	Ramirez - Fortnite character	5
2.1	Spiderman: Into the Spider-Verse	7
3.1	“The Boy Who Cried”	12
3.2	“Brave Player” Character Model in DPA-pipe	16
3.3	“Brave Player” Character Rig in DPA-pipe	17
3.4	“Brave Player” Character Animation in DPA-pipe	18
3.5	“Brave Player” Character render in DPA-pipe	18
3.6	“Brave Player” project on Shotgun	19
3.7	“Brave Player” project in Toolkit	20
3.8	“Brave Player” project publisher in Toolkit	21
3.9	“Brave Player” project loader in Toolkit	21
3.10	Disconnect project on Shotgun	26
3.11	Disconnect dog rig in Toolkit	27
3.12	Disconnect dog rig in Toolkit	28
3.13	“For A Rainy Day” game production in Toolkit	30
3.14	Shotgun Desktop	31
3.15	“For A Rainy Day” Coin Model	31
3.16	“For A Rainy Day” Coin FBX Publish	32
3.17	“For A Rainy Day” Custom Exporter	32
3.18	“For A Rainy Day” Play Test	33
3.19	Type of productions done by students	34
3.20	Production pipeline experience of students	35
3.21	Modeling tools used by students	35
3.22	DPA pipeline experiences of students	36
3.23	Texturing tools used by students	36
3.24	Renderer tools used by students	37
3.25	Shotgun Web Service experience of students	37
3.26	Animation tools used by students	38
3.27	Shotgun Toolkit experience of students	38
3.28	Game Development experience of students	39
3.29	Game Engine used by students	39
3.30	Student work locations	40
3.31	Compositing experience of students	40
4.1	Animation production workflow	43
4.2	Maya workflow	45
4.3	Game production workflow	46

5.1	DPA-pipe Shotgun Publisher	51
5.2	Tractor Render Queue Submission tool	53

Chapter 1

Introduction

Production is defined as “the action of making or manufacturing from components or raw materials”. In computer graphics, “*pipeline*” refers to the collection of scripts and processes that streamline passing data between programs and across departments [13]. An example of a production pipeline in manufacturing is building a car. Getting the raw materials, developing the parts, and finally assembling the car are its various production stages. This same principle applies in film making. A game development pipeline might differ from its motion picture counterpart, but they all have a standard shared workflow from which they originate.

Every film starts with an idea or a story. The story then goes through a series of refinements in a phase called *pre-production*. In pre-production, the story or idea goes through script writing, concept art, storyboarding, and pre-visualization. After the scripts and story boards have been approved, the next phase of film making is called *production*. Artists in different departments start working on creating characters, environments, and props generalized as *assets*. These assets are passed on to layout, texturing, and rigging artists.

The layout artist is responsible for creating “*shots*” described in the scripts from pre-production. The rigging artist creates the anatomical mechanics of the characters and the animation artist takes the “*rigs*” and creates movement in the shots from layout. Texturing and surfacing artists design the material qualities of the assets in a stage called look development. Look development starts at concept art where an artist would draw the general appearance of characters and props. After animation, the shots and surfacing materials are given to the lighting artist to create the “*mood*” of a scene. Some shots are also passed along to an Effects(FX) artist who specializes in

simulating cloth, water, magic, and character extras like human crowds. The final image is rendered from lighting or FX and composited together. A compositing artist is responsible for making final touches and fixing errors in the rendered images. After compositing, editorial takes all the shots from the previous artist, adds color corrections and finesses the shots to push the story with dialogue and music. The final product is an animated film that is shown in theaters and festivals. Figure 1.1 shows a traditional pipeline from Pixar Animation Studio [14].

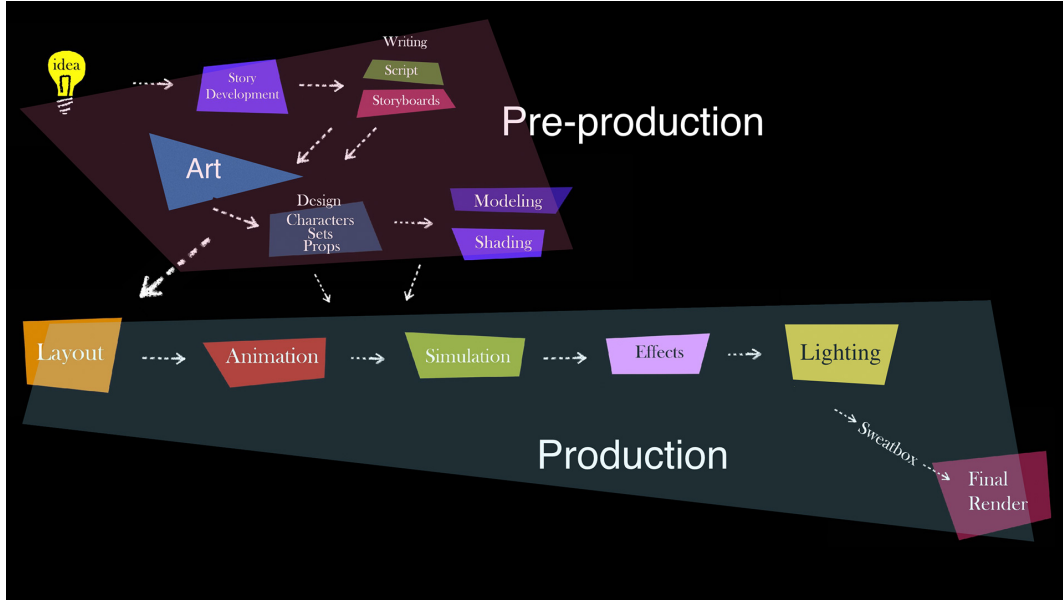


Figure 1.1: Example of a Traditional Pixar Pipeline

1.1 Asset Management

The production process can last as long as a few months to many years with teams of a dozen to hundreds of thousands of artists across many studios. To manage all the interactions between artists and data passed between them, a system had to be created to manage and track the chaos. Asset management is the process of documenting and tracking all assets in a production and making sure they remain connected and readily accessible [6]. Most studios have dependency graph systems that track assets and all their dependencies in the pipeline through meta-data embedded in published files. Dependency graphs can be visualized as a relationship between a parent and a child. Decisions made by the parent can affect his/her children. To put this in perspective, a scene

consists of a cube with a rig and texture. If the modeling artist changes the cube to a sphere, the rig and textures of the cube are affected by this modification. This makes modeling the parent with rigging and texturing as its children. Another example of a dependency graph system is Autodesk's Maya node editor. This editor keeps track of histories that make up the current asset in the form of a node graph. Figure 1.2 shows an example of a node graph for a simple Lambert shader.

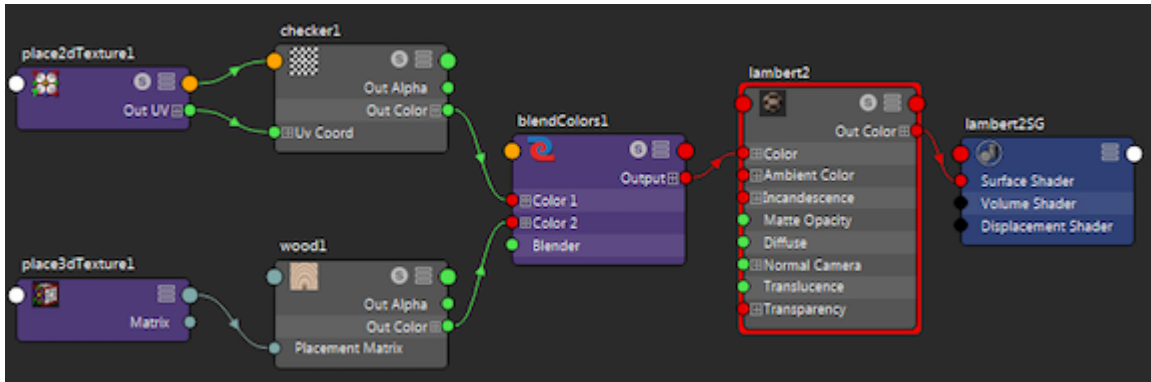


Figure 1.2: Example of Maya Shader Dependency Graph

Asset management systems can be as simple as folder structures or as elaborate as graphical user interfaces(GUI) that have access to a database of assets in a production. This forms a strong foundation for tracking and managing large or small projects.

1.2 Project Management

Project management is the practice of initiating, planning, executing, controlling, and closing the work of a team to achieve specific goals and meet specific success criteria at the specified time [20]. This system allows managers and department supervisors better assess the progress of their teams and prioritize and reassign tasks during bottlenecks in production. Artists are more efficient when these systems work effectively. Unfortunately, a project management system is only as good as the person who uses it.

Pipelines ensure each stage in the production is receiving the right data format whilst the asset management system keeps track of the data being transferred to each stage. Finally, the project management system makes sure each task is completed efficiently and on schedule. These three systems ensure the success of the artist and the project. Although pipelines are very beneficial

in productions, sometimes they can be a hindrance to efficiency and can create bottlenecks. A pipeline might automate several tasks an artist would take to perform an action but, more often than not, this task may take the automation tool several minutes to complete versus a couple of minutes for the artist. If this action needs to be performed multiple times in several steps down the pipeline, the pipeline becomes inefficient. Pipelines are only as good as the tools in it.

1.3 Art and Pipeline

“**Star Wars**” film Director, George Lucas once said, *“All art is dependent on technology because it’s a human endeavour, so even when you’re using charcoal on a wall or designed the proscenium arch, that’s technology.”* Technology in art empowers artists to push their creativity, explore new techniques, and produce incredible experiences in storytelling. Pipeline is a form of technology. Tools in production pipelines streamline tedious processes in productions giving the artist and directors the liberty to expand their creative ideas. Epic Games’ **“Kite”** Demo and **“Fortnite”** Launch trailer introduced the concept of using a game engine such as Unreal Engine, to make an animated short film. This process of film making is called *“Machinima.”* Using a game engine allows artists to create, iterate, and render the film in half the time it takes to make a traditional animated film. The game engine as their renderer - *“a software or hardware process that generates visual images from a model”*[22] allows artists to produce images at a maximum rendertime of **41.67 milliseconds** per frame compared to traditional animated film rendertime of a few minutes to several days. A pipeline tool used in the **“Fortnite”** trailer was a custom Filmbox(FBX) Alembic export format that gave animation artists the liberty to create high resolution facial animations. This tool utilizes all of Autodesk Maya’s deformations and rigging techniques with little no compromise in the game engine[2].



Figure 1.3: Ramirez - Fortnite character

When an artist dreams up an idea, the technology is created to support that idea. This technology sparks a new idea that brings about innovation. Pipelines fuel creative innovations in film making which bring new and captivating experiences to its audience.

Why use pipelines in production? Pipelines allow artist to iterate during productions with ease and efficiency. This enables artist to explore new ideas and focus on the content they are creating. Pipelines also enable project scalability by allowing artist to easily share content and data between themselves. This streamlined process reduces the burden of artists knowing the intricate details of the production process and helps them focus on what they enjoy doing.

For this thesis, I will explore ways of encouraging and promoting the artists creative processes through a study on previous production workflows in Clemson University's Digital Production Arts. This study will also give insight into the advantages and disadvantages of production pipelines in projects and how to better tailor pipelines to allow ease of use, efficient workflow and enhanced creative exploration for artists.

Chapter 2

Background

2.1 Industry Production Pipelines

Production pipelines in most studios vary from project to project. In 2014, Rhythm and Hues Studios(R&H) presented a pipeline solution the studio has used in its feature films since the year 2006. This pipeline was designed to support multiple locations and vendors. “The entire production pipeline is modeled as a dependency graph. Each node in the graph represents a work area - a production context where an artist does their work; for example modeling a pig, rigging a tiger, or animating a shot”[4]. During the production of **“Evan Almighty”**, the pipeline team streamlined the animation of several hero characters in a shot by creating multiple work-areas for animators with a few of the hero characters each scene. As the animators worked, they would export cached geometry from their scenes of which animators in the other work areas could subscribe to and see how their characters animated with the rest of the hero characters[3].

As production pipelines evolve, film makers and artists explore other ways of telling stories using tools in the pipeline. Pixar’s **“Incredibles 2”** director, Brad Bird, wanted to explore making **“Incredibles 2”** with a live-action film technique. Computer Generated(CG) cameras and lights had to mimic the behaviors of its real-life counterparts. Advances in the rendering pipeline allowed layout and lighting artists to employ their knowledge in photography to better push the visual fidelity of each shot in the film. This freedom for the director and artists gave the film its iconic cinematic and live action aesthetic[14].

A few productions go against the traditional workflow of film making and require the pipeline

team to make changes to accommodate them. “Spiderman: Into the Spider-Verse,” made by Sony ImageWorks, winner of Best Animated Feature Film 2019 Academy Awards, is a very good example of a visual style that required the production team to modify the pipeline. Figure 2.1 shows the visual style of “Spiderman: Into the Spider-Verse” [10].



Figure 2.1: Spiderman: Into the Spider-Verse

“Into the Spider-Verse” introduces a new Spiderman and a revolutionary visual style of animation to Sony Picture ImageWorks’ production pipeline. The production pushed the boundaries of traditional computer graphic animation pipelines so its creators could stay true to the inspiration of traditional comic-book storytelling. This technique required several department artists to invent new ways of animation and simulations [7][23].

Traditional computer graphic animations are created at 24-30 frames per second also known as real-time - which means there is an image for every single frame. With “Into the Spider-Verse,” the animators dropped 12 frames out of the 24 to achieve the visual style they wanted. Slowing down the motion of characters created a sort of staggered effect which mimicked traditional 2D animation styles from well known animators like Glen Keane. This change in animation broke the traditional way of simulation, lighting, and camera work. Motion blur, an important part of cinematography created by camera or character movement, was dropped in support of more traditional print style chromatic shifts. Simulations that require real-time motion to calculate physics on objects also had to be reinvented to pull off 12 frames per second[9]. Ghost frames in between animations on two’s had to be developed to allow the cloth simulations to calculate normally.

Production pipelines scale based on project needs but the fundamental workflow stays relatively the same. Small projects may not require as many resources as big budget films or AAA (pronounced “triple A”) games may need. A short film, about 2 minutes long, may only have a main character and a few supporting actors, props, and environment in the story. This can be done with as little as 1-5 artists. Bigger projects may have hundreds of characters, environment pieces, and props spanned across multiple shot sequences done by hundreds, if not thousands, of artists and technical artists. This workflow usually outputs gigabytes to even terabytes of data requiring a robust infrastructure to be in place to handle this immense amount of data.

2.1.1 Pipeline Infrastructure

Large Data facilities are a part of most large scale studio pipelines. An artist may create a single asset for an environment piece in a project. This asset would go through several iterations before it is finally approved to be the final representation of what the supervisor or director envisioned. These iterations are represented as versions in the pipeline workflow. Data facilities in studios are usually server based and shared across departments for easy data transfer and access. These data centers also have strict permissions and security protocols to avoid client information from “leaking” to the general public. This strategy allows the studio to control how they market their products to the public and clients and also prevents unauthorized users to make modifications to files.

2.2 DPA Pipeline

The DPA Pipeline also known as “DPA-pipe”, is a custom pipeline framework developed and lead by Dr. Jerry Tessendorf, Josh Tomlinson, and students in the Clemson University Digital Production Arts program with the intended purpose of encouraging student artists to use and learn pipelines in production. DPA-pipe’s architecture gets its inspiration from the successful Unified Pipeline infrastructure from Rhythm and Hues Studios. The core concept behind the pipeline was, artists do work in multiple locations, they use software, and each artist shares data. This concept influenced the hierarchical nature of the pipeline introduced as “*Production Task*” or “*pTask*”. “*pTask*” are abstract representations of tasks that need to be completed on production[1]. Each *pTask* is identified by a type that gives insight on the location within a production hierarchy. These types are: project, phase, build, shot, stage, work [1].

- **project:** This is the root level production task. All other pTask are represented under project. An example is, “The Boy Who Cried”.
- **phase:** Phases are split into production, pre-production, and research and development.
- **build:** As the name implies build can be categorized as the location for creating character, prop, and environment. An example is a “Boy” asset in “The Boy Who Cried” project.
- **shot:** A series of frames that run for an uninterrupted period of time[21].
- **stage:** “Defines the major work stage of a project”[5]. This can be animation, modeling, texturing and rigging.
- **work** This can be defined as a place where an artist does work. This is usually the workarea in a stage.

DPA-Pipe has been used in almost all major projects in Clemson University’s Digital Production Arts program since the year 2014. Productions like **“Peanut Butter Jelly”**, **Dream-Works Summer Projects: “Swept Up, Disposable”**, and many others each created and used tools available in the pipeline to help push storytelling and create a seamless experiences for artists. Tools like *dpa_subscribe* gives artists the ability to request and use assets created in other stages of the pipeline without searching for its physical location. The subscribed assets are also updated when a change is made upstream in the pipeline. Other tools developed in the pipeline such as *darkKnight* (pronounced Dark Knight), and *dpaffmpeg* (pronounced dpa f f m peg) allow artists to streamline tasks such as sending render jobs to a renderfarm and generating movie files for review from a sequence of frames. In **“Roboasis”**; a 2013 student film, the team developed several tools in the pipeline to help surfacing artists, animators, character Technical Directors, layout artists, and lighting artists streamline their work and focus more on the aesthetics. An animation tool developed during production allowed animators to move the main robot character in 3 dimensions while the tool took care of rotating the character’s four wheels. The animator was able to focus more on pushing the character’s expressions without worrying about a trivial task like keyframing wheel rotations. Although DPA-pipe has had many successful uses in several productions, some recent projects have not been able to utilize the pipeline and its tools due to legacy features. We will explore this in further details in the next chapters.

2.3 Shotgun Toolkit

Shotgun is a project management system that has been in the industry for over a decade. This management system is used by major studios such as, Weta Digital, Rodeo FX, Image Engine, and many others to track, schedule, and distribute task to artists[17]. Shotgun also serves as a revision tool that allows producers, supervisors, and clients to monitor, review and give feedback on artist work during productions. Shotgun is integrated with an asset management system called *“Toolkit”*. *“Toolkit”* is the bridge between Shotgun and asset management applications. It is based on the concept of, “identifying what it means to build pipeline components that are useful for studios of all sizes without being able to make assumptions about how those studios operate”[12]. This robust system has allowed many studios to integrate in-house pipeline tools and workflows with Toolkit. This product is one of the most popular pipeline solutions in the industry.

In Clemson University’s Digital Production Arts’ program, Shotgun has been used in several productions to manage, schedule, review, and track project progress from both Clemson University’s Main Campus and Charleston locations. Full integration with Shotgun and Toolkit has also been used in multiple productions including a recent game production in Unity 3D. The integration of project management and asset management allows students in productions to collaborate with each other through peer review sessions and shared work experiences. Toolkit allows default features in the pipeline to be extended to support unique workflows. In the game production **“For a Rainy Day”**, a FilmBox(FBX) feature was integrated into the publish application to allow Unity 3D compatible exports from Maya. This integration allowed the modeling artist to focus on creating the main characters for the game while the added feature in the publish application took care of preparing the asset for Unity. Toolkit has been and continues to be a robust pipeline for productions but due to its “one size fits all” nature, it requires a deep understanding of how the system works and how to integrate custom tools. This sometimes makes the pipeline difficult to use during short production scheduled time.

Both DPA-pipe and Toolkit are very robust pipelines, and in most cases, its features are similar. This makes it difficult to choose a pipeline solution to use and forces the production team to determine the resources of the project beforehand. In the next chapters, we will explore some of the use cases of both pipelines in production; how it was used, what worked and what did not work.

Chapter 3

Case Studies

3.1 “The Boy Who Cried”

“The Boy Who Cried” is a story about a boy who befriends a wolf but his mother does not approve of this friendship. The short film was produced in a DPA Production class with a team of 7 student artists. A remote client(the director) and commuter students made Shotgun Toolkit the pipeline of choice for the production. This decision allowed the director to review and critique production progress from a remote location. The distributed configuration in Toolkit also gave student artists the opportunity to collaborate and work from home. This was the first Digital Production Arts project that used the full Shotgun and Toolkit integration. The production team decided to explore a style that had not been used in previous animated shorts at the University such as, setting the characters in a world made of paper and narrated by a child.

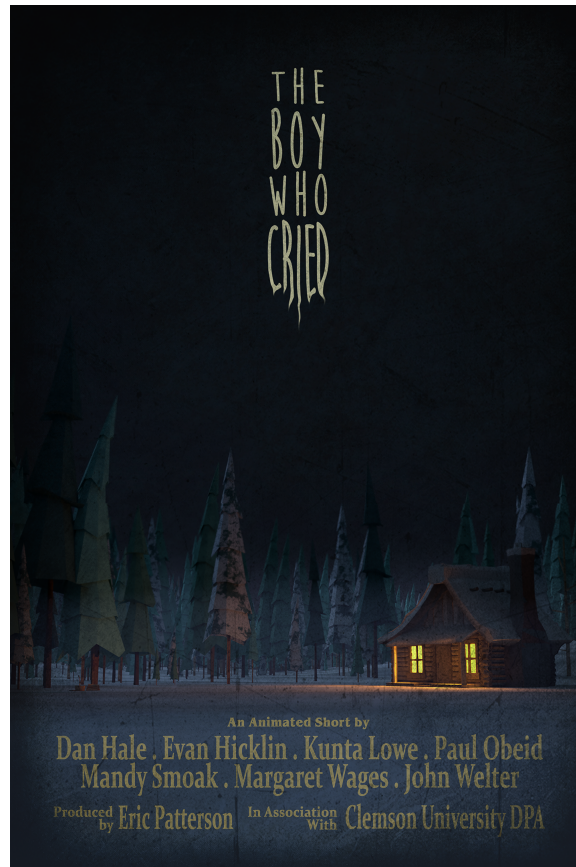


Figure 3.1: “The Boy Who Cried”

The production tools and pipeline used in this short film allowed the artists and creative directors to enhance the story by giving them the ability to use tools like the MASH node in Autodesk Maya. MASH allowed the artist to create real world scaled vast terrains - a pillar in the narrative of the film. This terrain enhanced the sense of vastness and loneliness of the character’s surroundings. With the help of the pipeline, artist were able to add fog, and push the lighting of each shots to give the story a stark and desolate appeal through a streamlined sharing process from the layout artist to the lighting and FX artist. This vision came with a set of challenges during production. In the next sections, we will explore the production and creative process of the film whilst looking at some of the challenges the artist faced during the production process in relation to pipelines.

3.1.1 Production

This film was produced with Shotgun Toolkit using the distributed configuration system. Each computer needed to have the configurations of the project stored in the Public Documents directory. This location allowed students to log into the campus computers and be able to access the project irrespective of their username and permissions. Autodesk Maya, DaVinci Resolve, RenderMan 21.4, Nuke, and Tractor were the softwares used in the production. The production workflow was **Modeling -> Rigging -> Animation -> Photography -> Surfacing -> 2D FX -> Lighting -> Rendering -> Compositing -> Editorial**. Listed below are the assets that were created for this project.

- Boy
- Mother
- Wolf
- Cabin
- Mountain
- 9 different Pine Trees
- Axe and Wood Pile

3.1.2 Process

This section describes the process used to make the short production in each pipeline. Each itemized section is a brief description of the process used in the various workflows.

- The team created all the assets, shots, and sequence list on Shotgun web and used the “create folders” command to setup their project directory on a shared drive,
- due to the visual style of the short, the modeling team had to design the characters as low polygon shapes with hard edges to match the feel of origami.

- After modeling, the layout team used the Autodesk Maya MASH tool to populate the environment with hundreds of trees. MASH also allowed the layout artist to control the location and scale of each individual tree.
- Rigging and surfacing were done in parallel. This allowed the lighting artist to start lighting downstream in the pipeline.
- After rigging, the layout artist placed cameras and characters in their start positions and passed on the characters to animation.
- Animation was done in each layout shot and then published to lighting.
- Due to the two main characters of the film having simulated clothing, after animation each character's garment was simulated in FX then later exported to lighting.
- After lighting, each shot was sent to a renderfarm for rendering and then published to the compositing and editing artist to add final touches to the film.

3.1.3 Success

This section describes features of the pipeline that helped in the production process. The itemized list is a brief description of how useful the feature was.

- Toolkit gave artists the opportunity to switch tasks when there were bottlenecks during production.
- Shotgun review in collaboration with Cinesync, allowed artists and supervisors to review and critique work from remote locations.
- Toolkit's multi operating system capability, allowed artists to work in environments with which they were most comfortable and use software that were not available on artists default operating systems.

3.1.4 Challenges

This section describes difficulties which were encountered during the production that related to the pipeline.

- Toolkit could not resolve files that were created on one operating system and opened on another. Referenced geometry in Maya was not auto-resolved when opened on operating systems that did not initially create it due to Maya’s absolute path system.
- Toolkit did not have an engine for DaVinci Resolve and Substance Painter. The exports for their software had to be moved and managed manually.

3.2 “Brave Player”

“**Brave Player**” is about a soccer player who meets his untimely demise while playing a soccer game. This short project was created with the intent of exploring the most current version of the DPA pipelines.

Since there were only two artists working on this project, it did not require a lot of resources. For this project in DPA-pipe, Autodesk Maya 2016, Nuke 11, and Renderman 21.4 were the software used in the production. In Toolkit, Autodesk Maya 2018, and Renderman 22.3 were used. Both pipelines followed the same workflow to avoid duplicate work. Assets made in one pipeline were exported and reconstructed in the other. Below is a list of assets in the short.

- Soccer Player
- Large Soccer Ball
- Small Soccer Ball
- Soccer Field
- Trees
- Fence Post
- Two Stadium Light Posts

The production pipelines (both DPA-pipe and Toolkit) enable the artist to create this short film from start to finish in less than two days. Due to the tools in the pipeline that allowed easy transfer of geometry and animation data, the artists were able to focus on the aesthetics of the production. In the next subsections, I will discuss the software, processes, and findings made in the pipeline during the production process.

3.2.1 Production

For DPA-pipe, the team initially used the “*anim-short*” pipeline template to create the structure of the project. This was later abandoned for a custom template since the team wanted to control the file structure of the project. The pipeline used was, **Modeling -> Rigging -> Animation -> Surfacing -> Lighting -> Rendering -> Editorial**, with **Editorial** completed in Nuke. Renderman was the chosen renderer as this was what the team was most familiar.

3.2.2 Process

3.2.2.1 DPA-pipe

- Modeling was done outside of the pipeline and later brought into the Maya scene
“Brave_player=preprod=player=model=hero=maya=model_player.ma”.

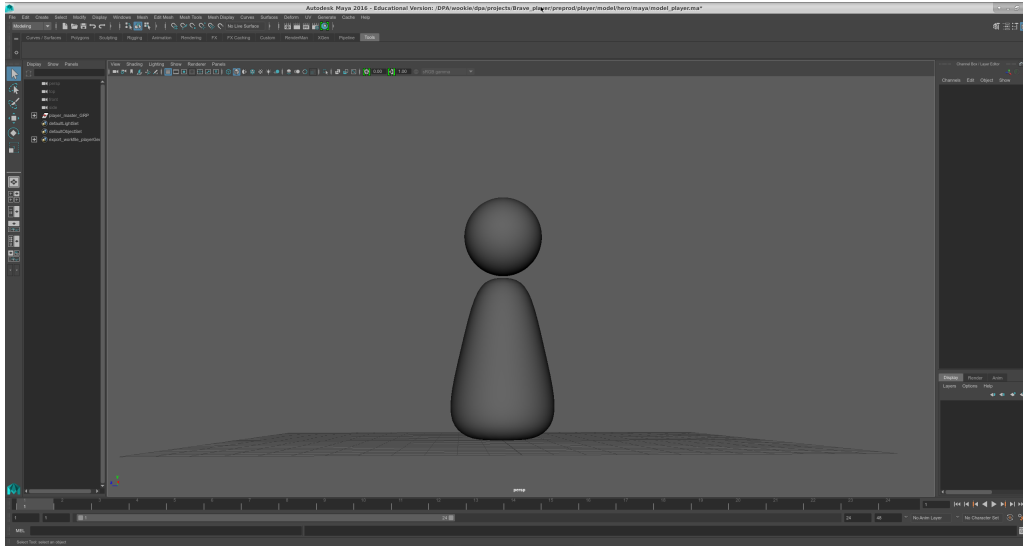


Figure 3.2: “Brave Player” Character Model in DPA-pipe

- After modeling, the geometry was exported out as an obj file, UV’ed in “Headus UV Layout” on Windows, reimported into the Maya scene, and replaced the original geometry with the UV’ed version for texturing later.
- The scene was cleaned up by freezing transforms, deleting history, and grouping the geometry under one top group. A Maya set was created as suggested by the pipeline documentation

called “export_workfile.masterPlayerGeo” for exporting the geometry to the next stage in the pipeline.

- The rigging artist then subscribed to the exported workfile and imported it into the rigging scene “Brave_player=preprod=player=rig=maya=rig_player.ma”
- The character was rigged with a single five-joint chain spine and bound to the geometry with five controls parented to the joints and exported to animation.

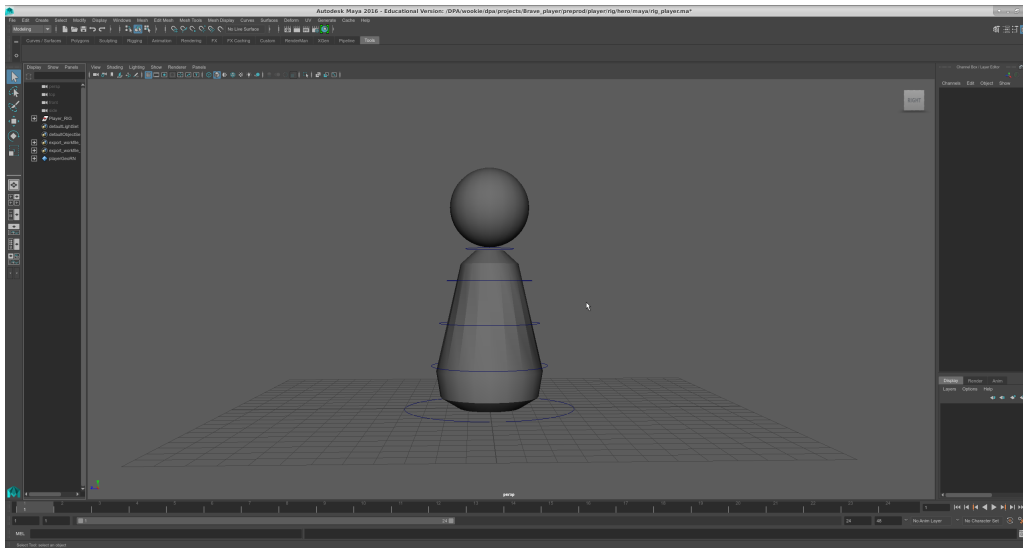


Figure 3.3: “Brave Player” Character Rig in DPA-pipe

- Animation was done in “Brave_player=prod=Go_Get_It=anim=hero=maya=Shot_01.ma” then the animated character was put in a set and exported to “Brave_player=prod=Go_Get_It=light=Go_Get_It_Light=maya=Shot_lighting.ma”

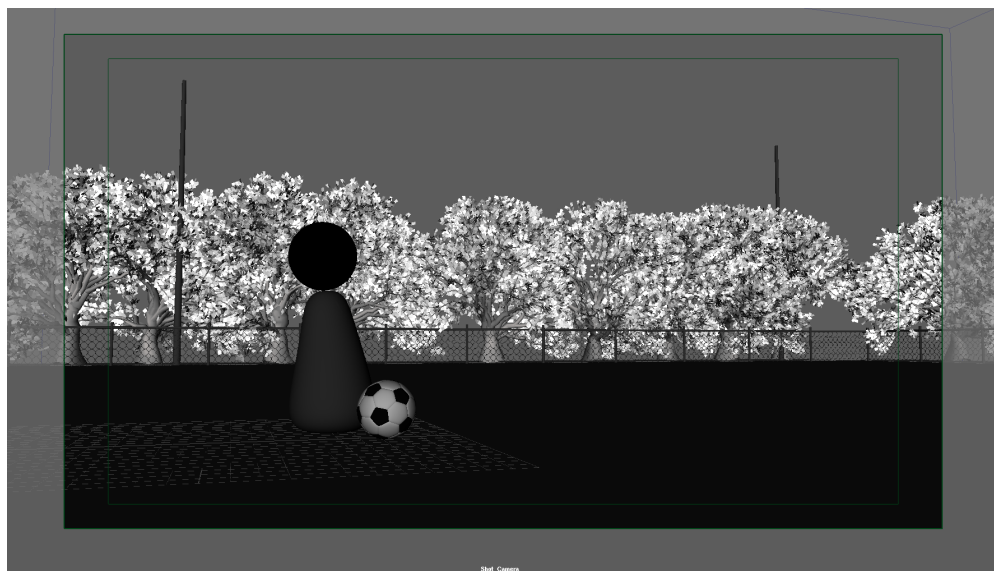


Figure 3.4: “Brave Player” Character Animation in DPA-pipe

- After surfacing, the lighting artist placed a few lights to match a nighttime soccer field scene and rendered through “*dark_knight.*”



Figure 3.5: “Brave Player” Character Model in DPA-pipe at 64 samples

3.2.2.2 Toolkit

- The team created a project on Shotgun called “Brave Player” and setup an asset list, shot list, and sequence.

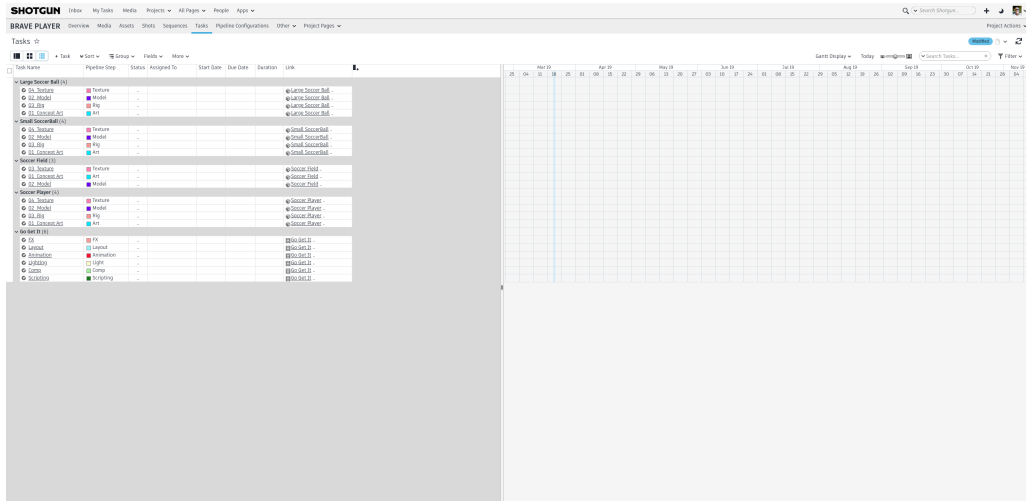


Figure 3.6: “Brave Player” project on Shotgun

- Shotgun desktop was installed, completing the project setup process.
- The default configuration name and distributed setup was used so the team could work on multiple operating systems.
- After the Toolkit setup was complete, the team created folders for the project through the create folders command on Shotgun.

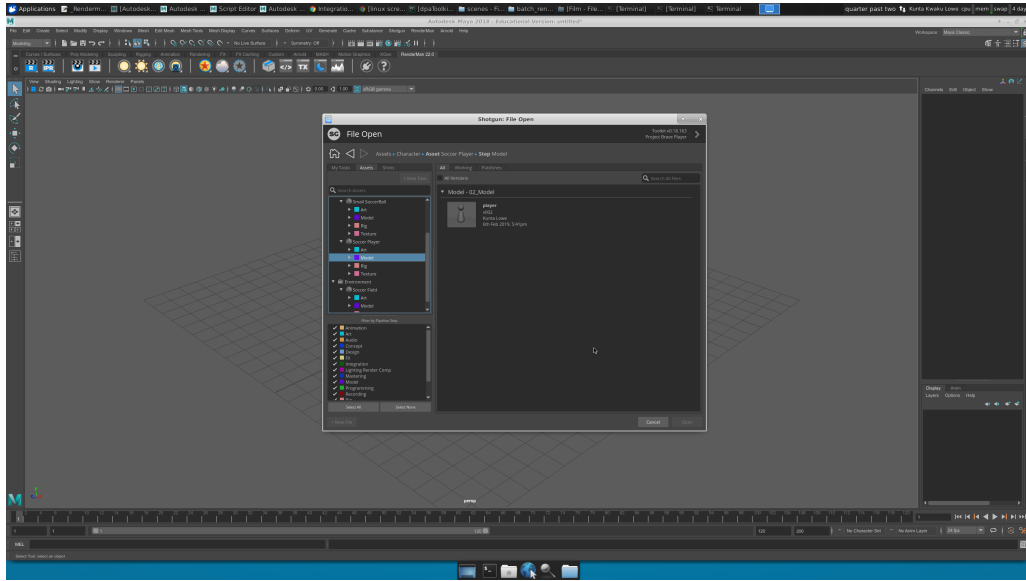


Figure 3.7: “Brave Player” project in Toolkit

- Modeling was done outside of Toolkit and brought into the respective scene in the pipeline.
- After modeling, the geometry was exported out as an obj file, UV’ed in “Headus UV Layout” on Windows, reimported into the Maya scene and replaced the original geometry with the UV’ed version for texturing later.
- The scene was cleaned up by freezing transforms, deleting history, and grouping the geometry under one top group. The file was saved through Toolkit and published using the built-in publisher.

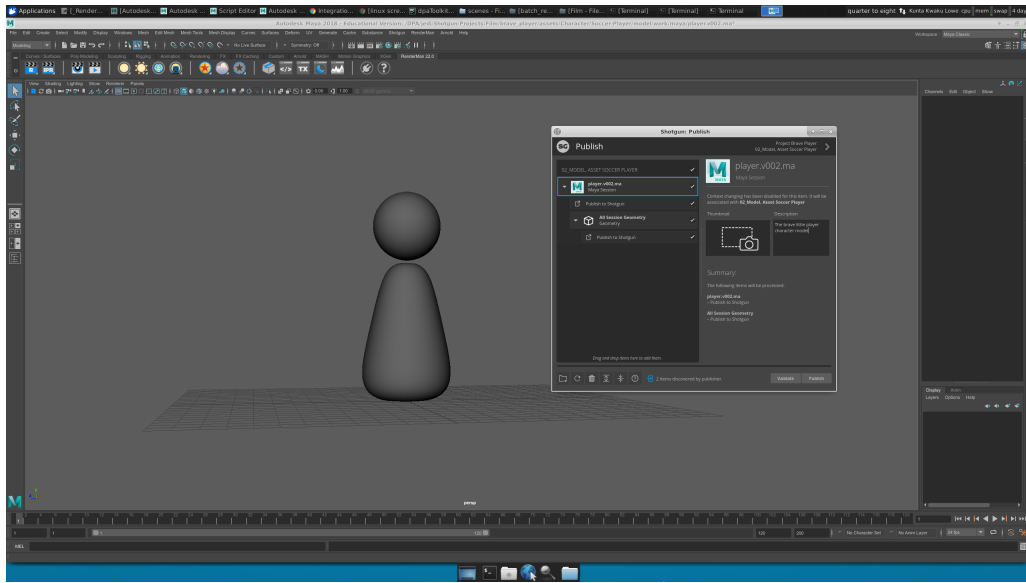


Figure 3.8: “Brave Player” project publisher in Toolkit

- Once assets were published, they were accessible via the loader app in Toolkit. The rigging artist used this to import the player model into rigging scene.

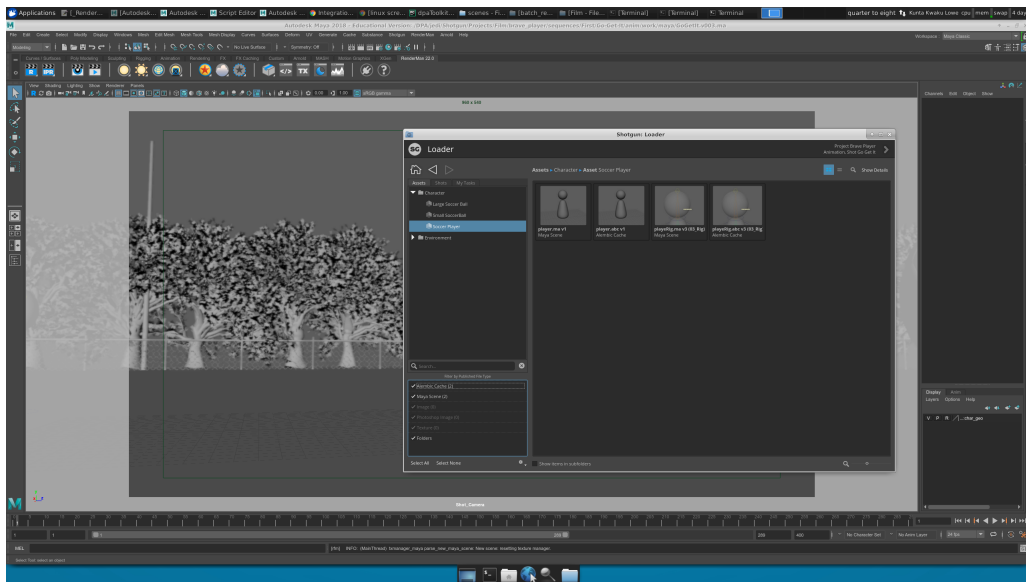


Figure 3.9: “Brave Player” project loader in Toolkit

- The character was rigged with a single five-joint chain spine and bound to the geometry with five controls parented to the joints and exported to animation. See Figure 3.2
- Animation was completed in the GoGetIt shot step and then published to the lighting step of the same shot.
- After surfacing, a few lights to match a nighttime soccer field were placed in the scene and rendered through the local queue as Toolkit does not have access to a native renderfarm.

3.2.3 Success

3.2.3.1 DPA-pipe

- Due to the hierarchical nature of the pipeline, you cannot create a ptask where it is not allowed.
- Exporting geometry and caches were very intuitive and simple(*Artist Workflow documentation*). You can selectively export specific caches.
- Pipeline seems highly customizable but strict enough to avoid project setup errors.
- Maya plugin supports shader transport. (This feature was not used in the production).
- The pipeline allows running surfacing and Rigging in parallel.
- A renderfarm plugin was available to render batch jobs from lighting.

3.2.3.2 Toolkit

- Toolkit has tutorials and documentation on how to navigate and use their interface.
- The pipeline is highly organized and setup in distinct categories, making file traversal easy.
- Native export of caches and scenes from asset step.
- The pipeline also recognizes playblast and rendered images during publishing.
- Toolkit allows artist sandboxing - which means artist are able to work on scenes that are not passed along to the next step.
- Supports Mac, Windows, and Linux.

- Mostly supports new and some Legacy versions of software.
- User interface was very straight forward and ties in well with Shotgun project management system without opening a browser.
- Allows setting up multi location configurations, thereby allowing outside sources to work on the same project.

3.2.4 Challenges

Some of the difficulties were due to first time user errors and would be noted.

3.2.4.1 DPA-pipe

- “pipeup” and “dpaset” commands were very slow
- Setting up the project took a long time to create due to the nature of the pipeline (**first time users**).
- Understanding the structure of setting up phases, builds, and stages were somewhat difficult. This process took a couple of tries but the team did not have enough time budgeted to figure out how to customize folder structures (**first time users**).
- The project manager on the team could not figure out how to delete a directory after it had been created, project, phase, or stage otherwise.
- The team could only use Maya 2016, a version two years behind the current Maya version. The idea of using XGEN for procedurally placing grass had to be abandoned and textures used instead.
- Mari did not open using the “dpa open” command so textures were done outside of the pipeline.
- “dpa open” was not setting the Maya projects properly. The team later learned to “dpaset” into a specific directory then use the Linux “cd” to keep moving down the file hierarchy(**first time users**).
- The team could not update subscriptions because the subscription process was made in the wrong directory(**first time users**).
- Batch render could not find the textures that were imported into the scene (**first time users**).

3.2.4.2 Toolkit

- Since texturing was done outside of the pipeline, the Publish2 app had to be used to bring in the textures but the textures would not show up in the loader app.
- The Publisher app in Maya exports the entire scene and does not allow selective publishes of elements, like animated characters to lighting instead of the entire layout with animation.
- Toolkit only exports the Maya scene and not an animation cache in the animation step.
- Toolkit does not allow switching OS's after referencing in Maya.
- Toolkit could not import a scene that had a reference.

3.2.5 Untested

This section describes features of the pipeline that were documented but not tested in this production.

3.2.5.1 DPA-pipe

- The team did not test texturing inside of the pipeline due to the software version of Mari.
- Compositing was not tested due to the software version of Nuke.
- FX as not tested in the short.
- Editorial step is currently not supported.

3.2.5.2 Toolkit

- Texturing was not tested inside of the pipeline due to the software of preference not supported(Substance).
- The team did not test FX as the short production did not require FX.
- Editorial software of choice is currently not supported (DaVinci Resolve). The supported software is not built for Ubuntu Linux.

3.3 “Disconnect”

“**Disconnect**” is a student short film project made by a Clemson University Digital Production Arts student, Daniel Hale, for his thesis. “This short film is an animated narrative critiquing our society’s increasing focus on communication through digital devices at the expense of traditional interactions. The film features stylized 3D animated characters composited with realistic 2D photographed environments to further accentuate the feeling of an augmented reality”, said Hale.

The Toolkit pipeline enabled Daniel to focus on creating the aesthetic appeal and message he was trying to portray in this short film. After rendering the film, he thought it would enhance the story if he added an extra shot with a closeup of his character to push the emotion in the film. He was able to easily create an extra scene with all the characters and camera setup he needed without compromise to the other shots by using the publish and loader tools available in the pipeline. In the next sections, we will explore Daniel’s production process and the challenges he faced during the making of “**Disconnect.**”

3.3.1 Production

This short film was produced with Shotgun Toolkit using the default configurations. To make this project portable, the configurations were customized to store workfiles and published data on Daniel’s personal mass storage drive. This allowed him to work from home as well as on campus with only being at the mercy of an internet connection and the read/write speed of his storage drive. Daniel used Autodesk Maya 2018, Arnold renderer, Adobe Photoshop, Headus UV Layout, Nuke, Adobe Lightroom and Adobe Premiere for the project. To reduce the amount of time it would take to produce the film, the assets were prebuilt by supporting artist outside of the pipeline. The workflow used on the production is, **Modeling -> Rigging -> Surfacing -> Lighting -> Animation -> More Lighting -> Rendering -> Compositing -> Editorial**. Below is a list of the assets in the short film.

- 2 Human characters
- 2 Dog characters
- 2 Cellphones props
- Stuffed Animal props

- Ice Cream Cone prop
- Sunglasses prop
- Projection Planes
- Proxy Geometry(shadow catchers)

In the next section we will discuss the process used in making this short film.

3.3.2 Process

- Daniel created a project on Shotgun called “Disconnect” and set up an asset list, shot list, and sequence.

Task Name	Pipeline Step	Status	Assigned To	Start Date	Due Date	Duration	Link
Bench (3)							
03_Texture	Texture	-					@Bench -
01_Concept.Art	Art	-					@Bench -
02_Model	Model	-					@Bench -
Engagement ring (3)							
03_Texture	Texture	-					@Engagement ring -
01_Concept.Art	Art	-					@Engagement ring -
02_Model	Model	-					@Engagement ring -
Ice Cream Cone (3)							
03_Texture	Texture	-					@Ice Cream Cone -
01_Concept.Art	Art	-					@Ice Cream Cone -
02_Model	Model	-					@Ice Cream Cone -
iPhone (3)							
03_Texture	Texture	-					@iPhone -
01_Concept.Art	Art	-					@iPhone -
02_Model	Model	-					@iPhone -
Mia (4)							
04_Texture	Texture	-					@Mia -
02_Model	Model	-					@Mia -
03_Rig	Rig	-					@Mia -
01_Concept.Art	Art	-					@Mia -
Mr. S (4)							
04_Texture	Texture	-					@Mr. S -
02_Model	Model	-					@Mr. S -
03_Rig	Rig	-					@Mr. S -
01_Concept.Art	Art	-					@Mr. S -
Queenie (4)							
04_Texture	Texture	-					@Queenie -
02_Model	Model	-					@Queenie -
03_Rig	Rig	-					@Queenie -
01_Concept.Art	Art	-					@Queenie -
Sean (4)							
04_Texture	Texture	-					@Sean -
02_Model	Model	-					@Sean -
03_Rig	Rig	-					@Sean -
01_Concept.Art	Art	-					@Sean -
Small Stuffed Animal (3)							
03_Texture	Texture	-					@Small Stuffed Animal -
01_Concept.Art	Art	-					@Small Stuffed Animal -
02_Model	Model	-					@Small Stuffed Animal -

Figure 3.10: Disconnect project on Shotgun

- He installed Shotgun desktop on his local machine completing the project setup process.
- The default configuration and distributed setup for Toolkit was used. This gave him the opportunity to work on multiple operating systems from different locations.
- After the Shotgun setup was complete, he created the folders for the project through the “create folders” command.

- Modeling and Rigging were done outside of Toolkit and brought into their respective scenes in the pipeline.
- After modeling, the geometry was exported out as an obj file, UV'ed in Headus UV Layout on Windows, reimported into the Maya scene and used to replace the original geometry's UV.
- He cleaned up the scene by freezing transforms, deleting history and grouping the geometry under one top group. The file was saved through Toolkit and published using their built-in publisher.
- Once assets were published, they were accessible by the Loader app in Toolkit. Since the characters were already rigged, Daniel went straight into animating his shots.

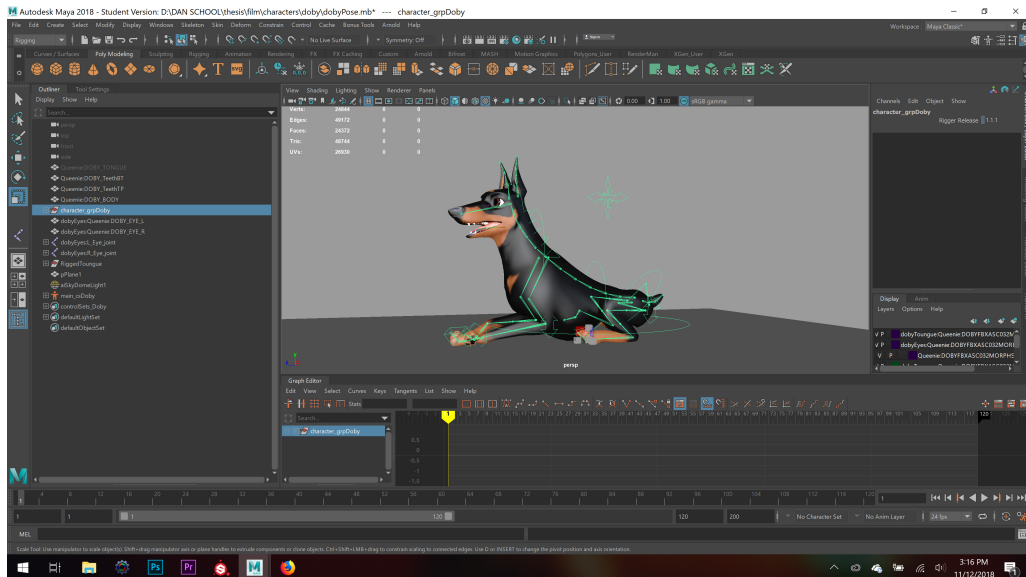


Figure 3.11: Disconnect dog rig in Toolkit

- Textures were made in Adobe Photoshop and published to the Maya scenes to be used for surfacing.
- After surfacing and lighting, the Toolkit scenes were rendered.
- After rendering, the rendered images were exported into Nuke, composited, and exported out as an EXR file format to Adobe Premiere for editing.



Figure 3.12: Disconnect dog rig in Toolkit

3.3.3 Success

- Toolkit helped him keep the project organized and aided in reviews with professors and collaborators on Shotgun.
- Due to the portability of the pipeline, Daniel was able to work in different locations - at home and on campus.
- Setup on a campus computer was simple during times a slightly more powerful computer was necessary.

3.3.4 Challenges

- The startup process were sometimes slow, usually when minor updates to a scene was all that was needed.
- Adobe Premiere was not supported in Toolkit.
- Not having a renderfarm access built into the pipeline made getting final frames difficult.

3.3.5 Untested

- The film did not require FX so that workflow process was skipped.

- Character models were pre-built, so modeling and rigging were not done in the pipeline.

3.4 “For A Rainy Day”

“For A Rainy Day” is a game about three children who are stuck inside their house due to bad weather outside. The children decide to put on a puppet show as a means of whiling away time. The player’s job, is to control the hero puppet. The hero’s objective is to stamp out evil. This is accomplished by collecting coins in the level, avoiding obstacles, and defeating the boss at the end with the current health of the player.

The pipeline tools in this production allowed the artist to focus on creating the mechanics and gameplay logic. Artists were able to export proxy geometry to the game engine to start the process of creating the mechanics and level design. During the course of the production, the level designers were able to increase the visual fidelity of the game by replacing the proxy geometry with its high resolution counterpart exported by the modeling and animation artist. In the next sections, we will explore the pipeline tools and processes used in the production in addition to the challenges the team faced with the pipeline tools.

3.4.1 Production

The production had a unique pipeline. It combined Shotgun Toolkit and Github version control system between the first and the last half of the pipeline. Shotgun and Toolkit were used for asset creation and tracking: modeling, rigging, animation and texturing. Github functioned as the version control tool for assets, levels, and scripts created in the game engine, Unity 3D. A custom exporter tool was developed to transfer assets made in Shotgun into the game project directory. The Unity project directory was created using a special script that contained information about the assets and game data that would be needed during production. Other software that were used in the production were: Autodesk Maya, Allegorithmic Substance Painter and Designer, Adobe Photoshop, Headus UV Layout, Zbrush, Topogun, and Google drive for documentation and task list summary.

Pipelines in game production are very complex as they require many iterations between asset creation and mechanics. The workflow used in this production followed a modified traditional workflow hierarchy. **Proxy_modeling -> proxy_rig -> placeholder_animation -> mechan-**

ics_development -> Low_resolution_modeling -> update_rig -> update_animation -> update_mechanics -> play_testing. This cycle of refinement continues until a high quality asset and game play is achieved. The following are the assets that were created for the game production:

- A Main Character
- “Imaginary world” Stage
- “Real world” Stage
- Rabbit
- Cannons
- Slope set pieces
- Dragon Boss

3.4.2 Process

- The project was created on Shotgun and Toolkit was setup with a custom configuration from Github. Git had to be installed on the artist computer to use Toolkit for the project.

GWWN	Overview	Assets	Levels	Animations	Tasks	Sprints	Notes	Published Files	Other v	Project Pages v		Project Overview
Tasks ☆												
<div> + </div> <div> Task ▾ Asset ▾ Pipeline Step ▾ Description Status Assigned To Due Date Duration Tags Link Filter </div>												
Game (10)												
> CS_Fly (6)												
> Narrative Script (1)												
Writer's Room <div>Concept - J. Caroline Requiere , J. Justin Weeks , J. Nina Foglesong</div> Game Narrative Script -												
PS_Heart Coin (3)												
Q1_Concept Art <div>Art - J. John Wester</div> Game PS_Heart Coin -												
Q2_Model <div>Model - J. Derek Andrews</div> Game PS_Heart Coin -												
Q3_Texture <div>Texture -</div> Game PS_Heart Coin -												
Level (35)												
> Desert (9)												
> Opening / Tutorial (8)												
White Box <div>Level Design -</div> Level Opening / Tutorial -												
Assemble Geometry <div>Level Art -</div> Level Opening / Tutorial -												
Asset Placement <div>Level Art -</div> Level Opening / Tutorial -												
Polish <div>Level Art -</div> Level Opening / Tutorial -												
Lighting <div>Lighting -</div> Level Opening / Tutorial -												
Seawen Placement <div>Gameplay -</div> Level Opening / Tutorial -												
Objective Scripting <div>Gameplay -</div> Level Opening / Tutorial -												
QA Gameplay <div>QA -</div> Level Opening / Tutorial -												
> Sea Cave (5)												
> Valley (9)												
White Box <div>Level Design - J. Derek Andrews , J. John Wester , J. Nina Foglesong , J. Mark Keller</div> Level Valley -												
Layout <div>Level Design - J. Derek Andrews , J. John Wester , J. Nina Foglesong , J. Mark Keller</div> Level Valley -												
Assemble Geometry <div>Level Art -</div> Level Valley -												
Asset Placement <div>Level Art - J. Derek Andrews , J. John Wester , J. Nina Foglesong , J. Mark Keller</div> Level Valley -												
Polish <div>Level Art -</div> Level Valley -												
Lighting <div>Lighting - J. Derek Andrews</div> Level Valley -												
Seawen Placement <div>Gameplay -</div> Level Valley -												
Objective Scripting <div>Gameplay -</div> Level Valley -												

Figure 3.13: “For A Rainy Day” game production in Toolkit

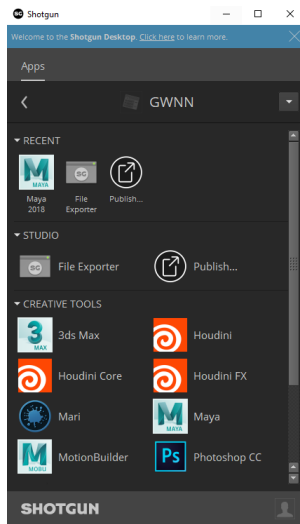


Figure 3.14: Shotgun desktop

- A custom script was used to create the unity project directory
- Proxy geometries were created outside of Toolkit then later brought into the pipeline through the builtin publish application.

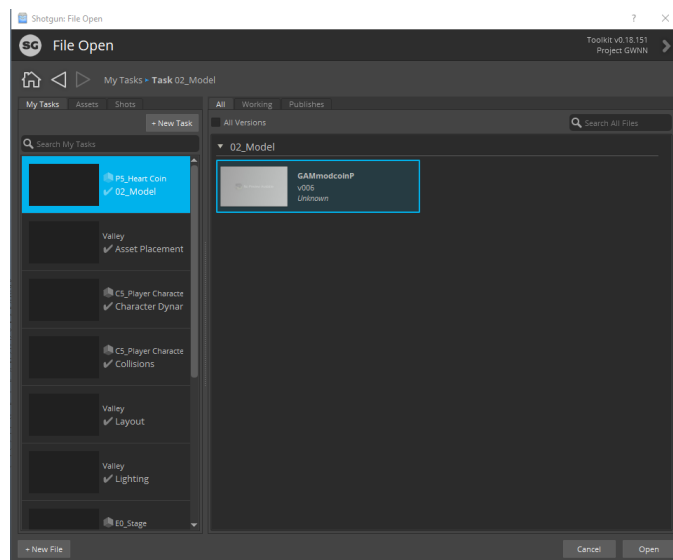


Figure 3.15: “For A Rainy Day” Coin Model

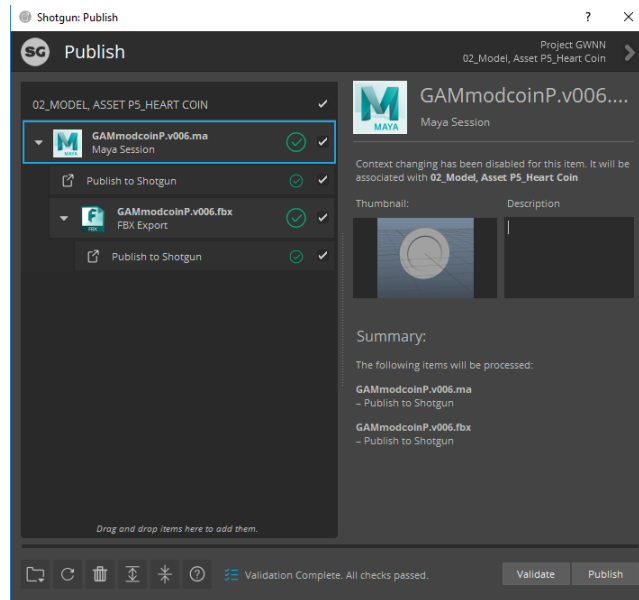


Figure 3.16: “For A Rainy Day” Coin FBX Publish

- Texturing was also done outside the pipeline and later published into the pipeline.
- After rigging and animation, a custom exporter tool was used to move assets from a Toolkit publish directory to the Unity project directory.

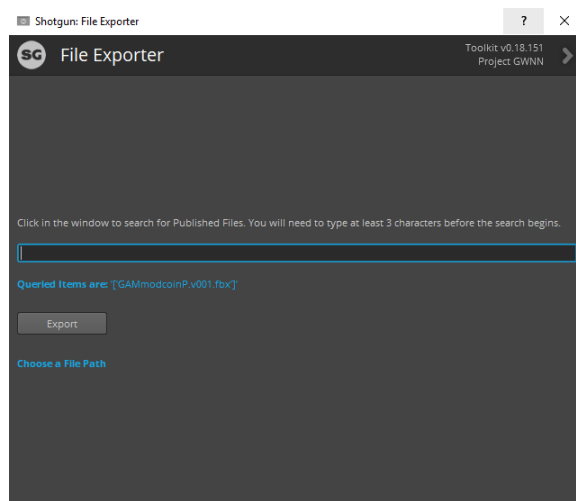


Figure 3.17: “For A Rainy Day” Custom Exporter

- After play testing in Unity, high resolution assets were exported from Toolkit and imported into Unity, replacing the proxy geometry.
- Since the game is in a real time engine, the final playable content was built and released through Unity.



Figure 3.18: “For A Rainy Day” Play Test

3.4.3 Success

- Centralized storage location allowed artists to collaborate on the project effectively.
- Shotgun made project management more efficient by being able to assign tasks to artists and view the progress of the project.
- Version control in the pipeline allowed artists to roll back mistakes or items that were accidentally deleted.

3.4.4 Challenges

- The functionality of Shotgun and Toolkit were too broad for the needs of the production.
- The extra step of using the exporter became a bottleneck when moving assets from one project location to the other.

- Shotgun Task Management system did not support creating task that weren't linked to assets.
- Artists' knowledge of the pipeline were limited.
- Toolkit was very strict on naming conventions, which did not allow special characters. This unexpected behaviour did not give artists the opportunity to be more descriptive in naming their assets.

3.4.5 Untested

- Editorial was not needed for the project since all the cut scenes were done in the game engine.

3.5 Statistics

A survey was conducted across Clemson University Digital Production Arts graduate students on both main campus and Charleston about production experience and their understanding of pipeline. There were 11 total responses. The charts below were generated from student surveys.

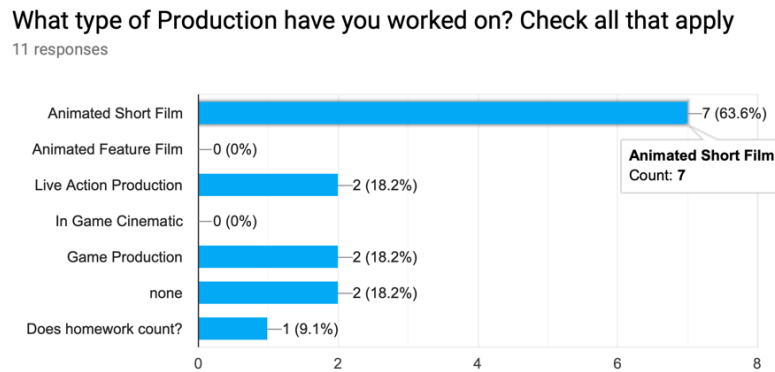


Figure 3.19: Type of productions done by students

Have you used a Production Pipeline before?

11 responses

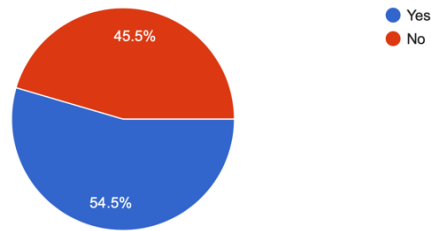


Figure 3.20: Production pipeline experience of students

What software do you use for Modeling? Check all that apply

11 responses

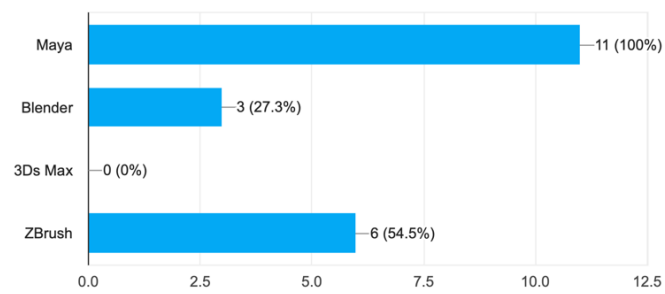


Figure 3.21: Modeling tools used by students

Have you used the DPA pipeline before?

11 responses

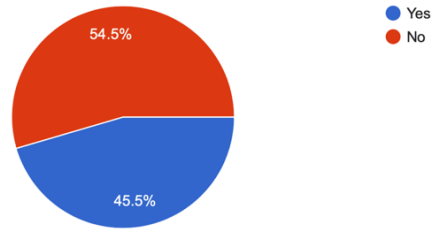


Figure 3.22: DPA pipeline experiences of students

What software do you use for Texturing? Check all that apply

11 responses

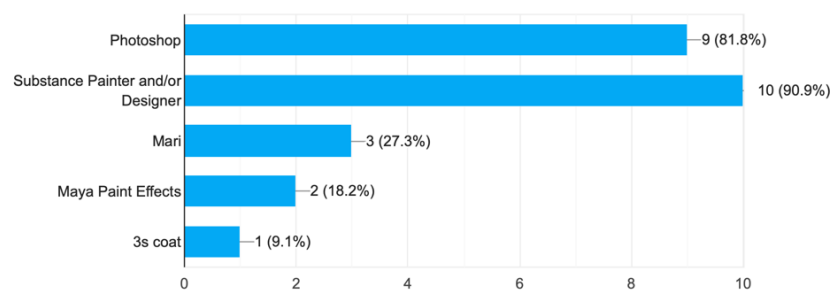


Figure 3.23: Texturing tools used by students

What Renderer and software do you use for Look Development and Lighting? Check all that apply

11 responses

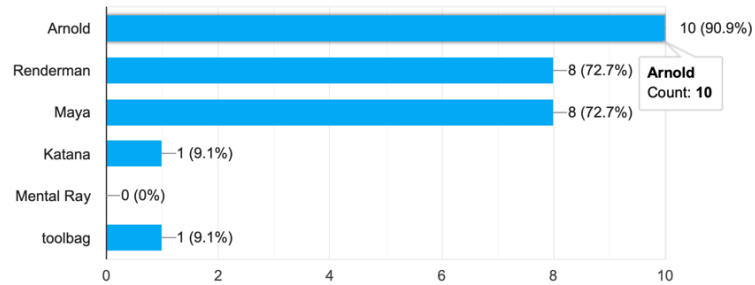


Figure 3.24: Renderer tools used by students

Have you used Shotgun Web Project Manager before?

11 responses

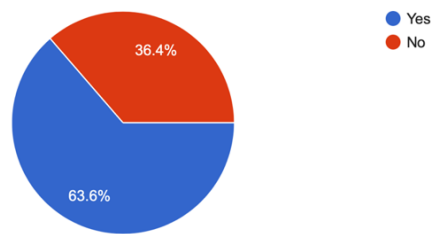


Figure 3.25: Shotgun Web Service experience of students

What software do you use for animation? Check all that apply

11 responses

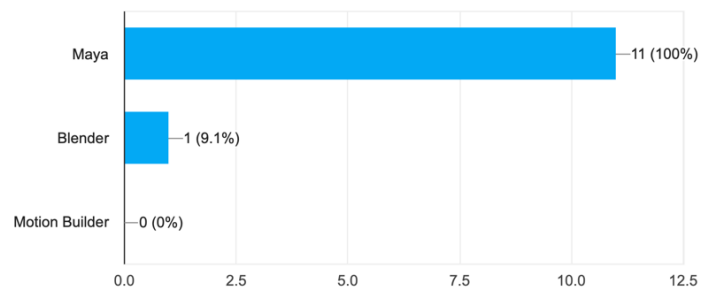


Figure 3.26: Animation tools used by students

Have you used Shotgun Toolkit before?

11 responses

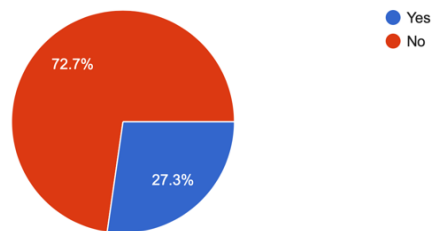


Figure 3.27: Shotgun Toolkit experience of students

Do you have a game development production experience?

11 responses

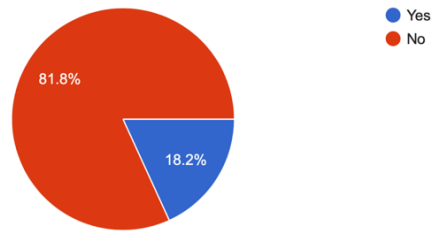


Figure 3.28: Game Development experience of students

What game engine have you used for game productions?

9 responses

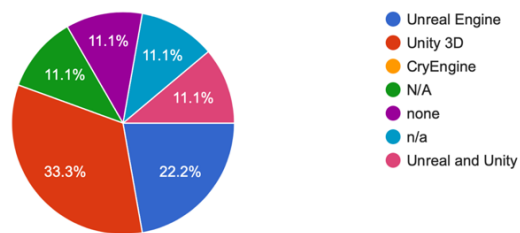


Figure 3.29: Game Engine used by students

Where do you mostly do your work?

11 responses

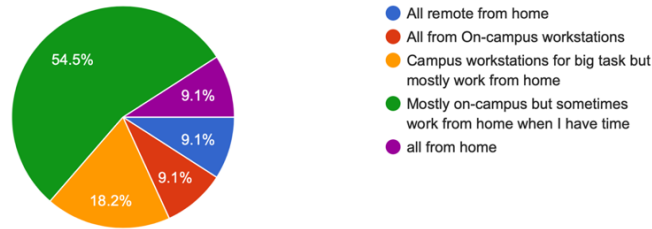


Figure 3.30: Student work locations

What software do you use for Compositing? Check all that apply

11 responses

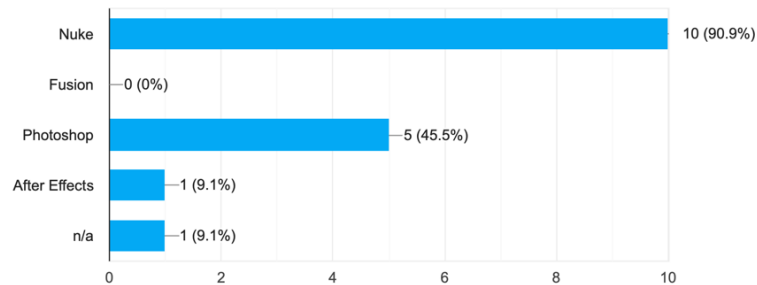


Figure 3.31: Compositing experience of students

The results above queried from student artists expresses the current state of Digital Production Arts' students knowledge and use of production pipelines. See Appendix A for the questions asked in the survey. Without prior knowledge of its workflow, students spend most of the production time learning how to use the pipeline and its application rather than focusing on the artistic endeavors of the production. In the next chapters, we will look at proposed workflows and tools that serve as templates or starting points for student productions. These workflows will enable students to focus more on the art of the production as well as learn the production pipeline process. The pipeline tools also proposed in the next chapter will enable students to use software they are most familiar with, learn new ones and gain experience with production and asset management systems.

Chapter 4

Design

Pipelines support artists by automating tedious processes and streamline hand-off procedures between artists but, sometimes it can create bottlenecks when tools break or custom features for a production are not supported in the pipeline. In Fall 2017, students in the DPA program in Charleston worked on a short film called **“Phooled”** for a production class. The production team planned to use Shotgun as their project management system and DPA-pipe as their asset management pipeline system. DPA-pipe was abandoned halfway through the production process because of an unsupported version of Maya in the pipeline that was necessary for the success of the production. The students had to use google drive as an alternative to the pipeline. This change jeopardized artist efficiency which pushed production past its intended deadline. **“Phooled”** was an example of a pipeline having a negative effect on a production.

In this section, three template workflows are proposed for various productions with feature updates to the current pipelines to improve efficiency, scalability, and usability based on the survey from DPA students.

4.1 Workflow

Pipelines are “living” structures that are intended to grow and evolve. It is designed to adapt to changing requirements of various productions whilst catering to the needs of the artist that use it. A dynamic pipeline allows projects to scale on a magnitude of artist and tasks required for a particular production. This also allows several projects to be ran concurrently without barely any

compromise from each other. At the core of the pipeline is the workflow. Workflows determine the interactions between steps in a pipeline. A pipeline step can be summarized as a specialized task within a production. Most art and technical teams in studios are categorized into these pipeline steps: Modeling, surfacing, and animation. The other steps in the pipeline are split up into specialized departments with sometimes internal pipelines between the artists and technical directors in the department. Workflows help these departments share data between themselves more efficiently and gives a clear structure on the steps in a pipeline (**Note:** This workflow assumes Research and Development(R&D) has been done for the project in pre-production).

In the next sections, a workflow is proposed for animation and live action short films with diverse tool integration for the steps in the pipeline. Additionally, a Maya based workflow is also proposed for the aforementioned films. And lastly, a workflow for real-time film and game productions is suggested. These proposed workflows are intended to serve as templates for various production but do not serve as a “one size fits all” workflow. Projects may use parts, all, or none of the workflow steps proposed.

4.1.1 Diverse Tool Workflow

This workflow includes popular specialized industry tools that are licensed by Clemson University and can be supported by both the DPA Pipeline and Toolkit Pipeline. This workflow covers a broad part of the pipeline but is dependent of the needs of the project. A project might depend heavily on lighting but not FX or vice versa. Figure 4.1 shows a 3D based diverse tool workflow.

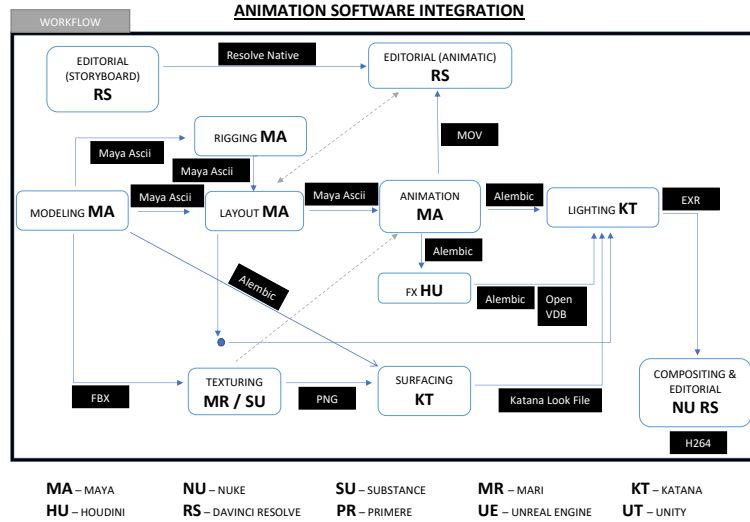


Figure 4.1: Animation production workflow

Starting at the modeling, the next steps in the pipeline can be paralleled with little to no bottlenecks since the steps may not be dependent on each other till later down the pipeline. Since animation is going to be done in Maya, a native Maya ascii file type is exported to the rigging and layout steps in the pipeline to prevent data loss. An FBX file type is exported to the texturing step. FBX is a high level export format supported by Autodesk and many third party texturing software like Mari and Allegorithmic Substance. The FBX format allows storing geometry, material information, cameras, and animations in one file. During texturing, rigging and layout may be completed and shots are created for animation. Keeping the native ascii file between rigging, layout, and animation ensures custom nodes made in rigging move seamlessly between layout and animation.

After texturing, a low resolution version of textures is shared with animation whilst the high resolution texture is passed on to surfacing. The textures shared with animation allow animators to analyze the impact of geometry deformation on textures. Maya supports several image file formats but, for this workflow, Portable Network Graphics(PNG) is suggested. PNG is a lossless compressed format which is good for getting high quality images at smaller file sizes than formats like Tagged Image File Format(TIFF). This format also allows alpha channels for transparency in textures.

After animation, an alembic geometry cache is exported from the Maya scene for lighting. Alembic caches like the FBX, are popular data sharing formats supported by many third party

software, in this case Katana. They allow storing cameras and geometry information with animation. This cache also supports the transfer of animation data to FX. Final frames can be rendered out of FX and composited later, but let us assume a cloth simulation was made so we will export an alembic cache into lighting.

Katana, developed by Sony Imageworks and later purchased by The Foundry, is a specialized tool for surfacing and lighting. It allows the surfacing artists to share data with the lighting artists seamlessly with a native file type called a *“Lookfile”*. Lookfiles allow surfacing artists to setup surfacing details about a geometry and share it with the lighting artist. The lighting artist only needs to import the Lookfile and resolve it with the built-in resolve node. This process speeds up lighting with an added bonus of geometries auto resolving to its shaders.

Final images are rendered out of lighting and/or FX as EXR file formats and passed along to compositing and editorial. Editorial runs parallel to the entire pipeline starting at storyboards in pre-production. They maintain the director’s vision of the whole story through a comprehension of all the sequences and shots made from storyboard, previzualization, animation, compositing, and sound. For this workflow, we will use DaVinci Resolve. Resolve is a popular industry nonlinear editing software that allows the editorial team to edit, color grade, and sound mix a sequence of images or movies. The final export format from Editorial is dependent on the needs of the project or client.

All the software and tools mentioned in the workflow above have public Application Programming Interface(API) that allows integration with pipeline tools. These software are also supported on the three major operating systems: Windows, MacOS, and Linux.

4.1.2 Maya Workflow

Similar to the Diverse Tools setup, the Maya Workflow follows the same hierarchy and data sharing model. The difference is in the software application used in the various steps. This workflow is designed around an Autodesk Maya pipeline. Maya would be used for modeling, rigging, layout, animation, FX, and lighting. The setup is designed for students who do not have access to commercial software like Katana, Houdini, and Nuke but would like to work on high quality productions. DaVinci Resolve, as of version 15, has a built-in compositing software called Fusion that is rival to Nuke. Maya and Resolve allow students to create high quality projects with minimal software data transcoding. Figure 4.2 shows a workflow with Maya and DaVinci Resolve.

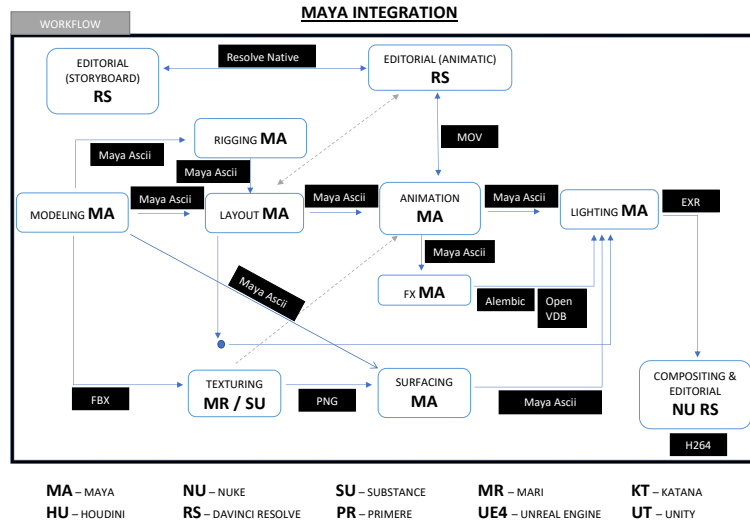


Figure 4.2: Maya workflow

4.1.3 Real-Time Engine Workflow

The Real-Time Workflow is designed for game and cinematic productions that use real-time rendering. Similar to the previously mentioned workflows, modeling is done in Maya and exported as an FBX format to texturing and the game engine for level design. Because of FBX's wide use and robust data format, it is also the main import file type for real-time engines like Unity and Unreal Engine. Pipelines in game engines are very complex. A game may depend heavily on mechanics which requires only proxy geometry and a high level of in engine testing. Another project may require simple mechanics but emphasis on the visual fidelity of assets and materials. This requires several iterations of assets and textures on the front end and very little time spent in engine. The most important component of this workflow is version control. Version control is a system that records changes to a file. This system helps recover unexpected catastrophes by allowing the artists and technical teams to roll back to previous versions of assets or "states" of work. The most popular version control systems are Git, Subversion, Mercurial, and Perforce. Most real-time rendering engines have plugins for the version control systems that not only store the state of items, but also stores a snapshot of the engine configuration. In bigger productions, these configurations are excluded in the version control since every artist in a production may have their own working

environment. Excluding these configurations reduces possible conflicts and data storage size on the server.

This workflow only gives a broad overview of the workflow for a real-time rendering engine. Figure 4.3 shows the workflow of a game and cinematic workflow in a real-time rendering engine.

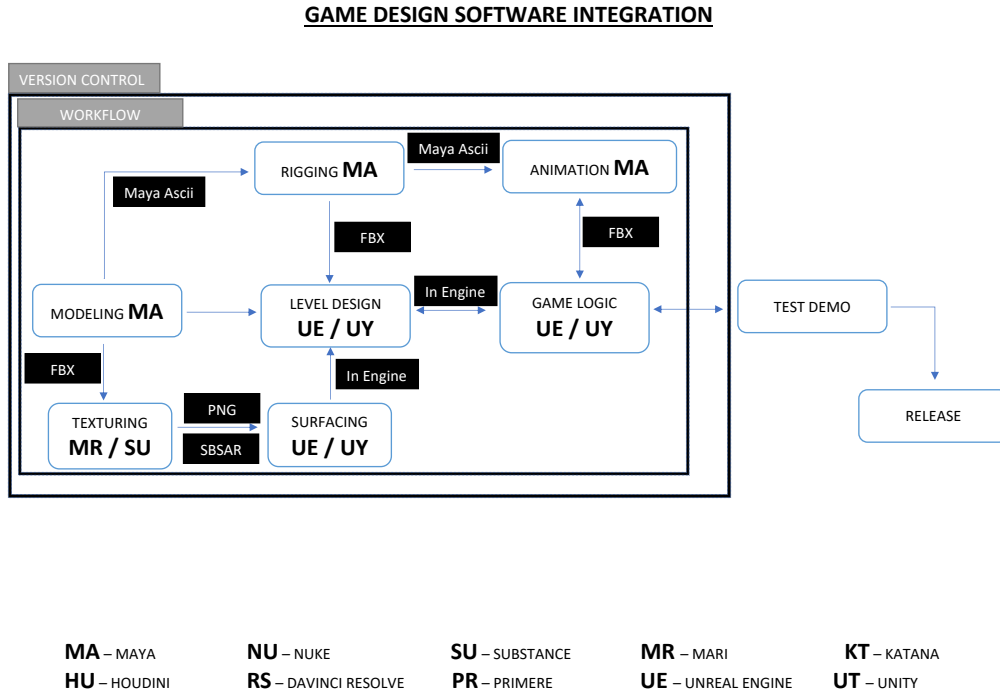


Figure 4.3: Game production workflow

Next, updates and upgrades to the current DPA pipeline are proposed to support current industry tools that boost efficiency and inclusion.

4.2 DPA-pipe Proposal

The DPA-pipe has been one of the cornerstone projects of the program since 2014. The pipeline has been used in several productions over the years and has benefited many students in getting experience working on and in a production. Although the DPA-pipe supports certain needs of a production, there are features that can be added that would be beneficial to current and next

generation productions. Listed are the features that would help improve the pipeline.

- **Interface with Shotgun project management:**

Shotgun Web is a robust project management interface that supports asset, task, and project tracking. It also allows applications outside of the web interface to connect and query information through a REST API. Once a project is created on shotgun, a command for DPA-pipe can be written to query project information and used to automate local file system creation. This process reduces the time it takes for production setup by 50%. This also allows an artist to create a project and start working in a short amount of time without the intervention of a pipeline technical artist.

- **Plugins for Substance Engines, Katana, and DaVinci Resolve:** The Substance Engines are a suite of texturing tools widely used in the industry. They speed up the process of authoring textures for assets by giving artists a real-time view of the materials. It also generates export maps that can be used in other render engines for high quality images. A plugin for Substance would allow artists to quickly create and export textures for their assets. These exports can then be accessible downstream in surfacing and lighting.

As previously mentioned, Katana is a lighting software designed to speed up the process of lighting sequences and shots. This software gives artists the ability to light multiple shots in a sequence at once whilst giving the artist control over each shot lighting through overrides. An application or plugin for the pipeline within this software allows all the previous steps to supply lighting with products non-destructively while speeding up the process of getting final images rendered.

A plugin for DaVinci Resolve allows final images from lighting and compositing to be dynamically loaded in its editing system. Resolve also uses the concept of databasing to give artists the ability to collaborate and share tasks without moving data or reloading settings.

- **Tractor Render Manager:** Tractor is a render manager developed by the Pixar team used to render and process batch jobs across networked computers. Tractor's robust scalability allows batch jobs to be processed in parallel or serially. Artists can process rendering tasks, composited images, and FX simulations at the same time from multiple locations.
- **Offload rendering to an independent renderfarm:** Rendering is the most resource in-

tensive process in the pipeline. Most rendering jobs are configured to use all the resources of a computer, CPU, and memory to produce an image. Using artist workstations not only slows artists down during deadlines but also reduces the life of the workstation. Offloading render intensive jobs to a farm like the Palmetto cluster, allows artists to work efficiently during close deadlines, reduces the cost of maintenance on workstations, and rendered frames are returned quicker due to the availability of computer resources.

- **Multi-platform load and publish assets:** Currently, DPA-pipe is only supported on the Linux operating system. This feature excludes art softwares like Adobe Photoshop and Zbrush. An application that allows artist to load assets into a software not native to the pipeline, gives the artist the ability to explore options when a problem arises. This applications will allow artists to publish assets to the right task or step without compromise to the workflow of the pipeline.

4.3 Toolkit Proposal

- **Launch Toolkit on Ubuntu:** Currently, Toolkit is supported on Windows, MacOS and Redhat based Linux operating systems like CentOS and Fedora, but DPA, being School of Computing, only supports Ubuntu Linux. Adding a tool to launch Toolkit on Ubuntu, gives artists the opportunity to access renderfarm resources that are currently only available on the DPA School of Computing workstations. This also reduces transfer and data decoding between operating systems since the software native file formats are irrespective of the system running them.
- **Add Engines for substance, Katana, and DaVinci Resolve:** Similar to DPA-pipe, a Katana and Resolve engine would be beneficial in creating a more streamlined workflow.
- **Add Tractor Batch Manager to Toolkit:** Currently, Toolkit does not have a built-in plugin for a renderfarm manager. Adding tractor to the Toolkit pipeline will allow artists to submit batch jobs from steps within the pipeline and get multiple results back.
- **Support export caches from animations:** Animation caches help reduce data size and information that are not needed or supported in the subsequent steps. This process allows faster scene load times in lighting or FX. Toolkit exports static alembic caches from Maya in

the asset creation phase but currently does not support caches from the shot phase similar to DPA-pipe. Adding this feature will make shots from animations made in Maya, transferrable to a simulation software like Houdini or a lighting software like Katana.

- **Support Shader and Camera exports:** Adding Shader and Camera exports to Toolkit, allows access to camera data from layout downstream in the pipeline without exporting work-files. Compositing artists will benefit from this feature when a reference camera is needed in a shot after the rendering process.

Chapter 5

Implementation

Based on the pipeline proposal from the previous chapter, three items were chosen, developed, and implemented as proof of concept. The tools were designed as templates that can be modified and implemented as plugins for DPA-pipe and Toolkit. First, a custom publisher that was created for artist to upload rendered images and movie files to Shotgun for review from DPA-pipe. This application would be similar to the builtin functionality in Toolkit. Second, a render submission tool was created to allow artists to submit render jobs from Maya to Tractor render manager. The submission tool has been modified to support other renderers like Arnold. The default Tractor submission tool that comes with Renderman for Maya only supports Renderman renderer. Lastly, a shell script was created to launch Toolkit on Ubuntu Linux.

5.1 DPA-pipe Shotgun Publisher

Revision is a fundamental task in production that allows artists and production supervisors to critique and give feedback on a project's progress. Although Shotgun is mainly used for project management, it also serves as a revision tool when integrated with a pipeline. To integrate DPA-pipe with Shotgun Web, a publisher tool was developed using the Shotgun Python Application Programming Interface(API) [18]. This publisher gives artists the opportunity to submit rendered images or movie files for review. The Application Programming Interface(API) contains modules for querying project data and uploading and downloading media. Publishing uploads media to a Shotgun server under a specific attribute like a project or asset. To make the publisher user friendly,

it was designed using QT. QT is a cross platform widget toolkit used in creating graphical user interfaces. Using this widget toolkit allowed the publisher to interface with Autodesk Maya since Maya’s graphical interface is also built on this platform. Figure 5.1 shows the custom publisher developed for DPA-pipe.

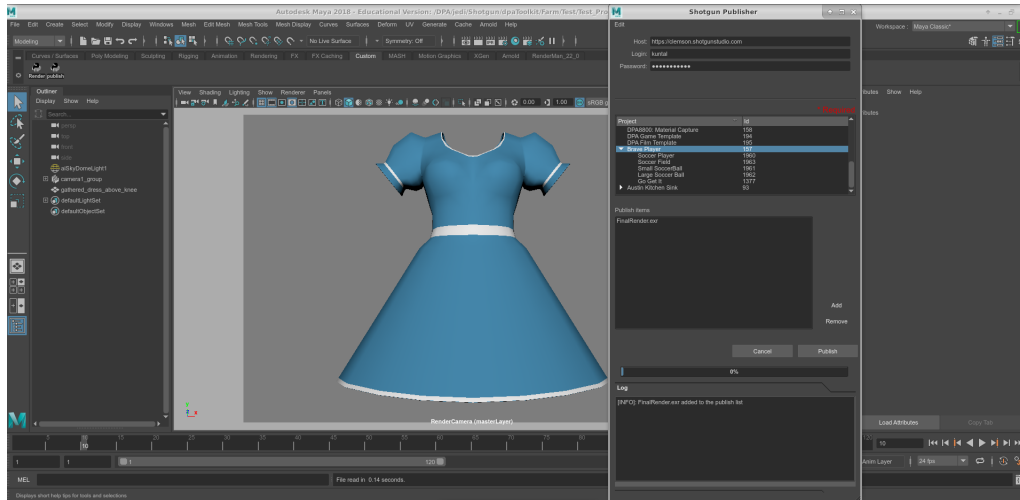


Figure 5.1: DPA-pipe Shotgun Publisher

The host, login, and password fields authenticates the user with the Shotgun Web interface. A successful authentication triggers a query event and populates the project window with projects listed on the site filtered by the username in the login field. Each project has an auto generated tag on Shotgun called an ID. The ID is unique to every piece of data on Shotgun. This allows pinpoint accuracy in filtering large amounts of data stored in the Shotgun database. On the publisher, each project in the project window has a drop-down list of assets and shots associated with it. During a publish, each item from the list is uploaded to the selected object in the project window. The add and remove options give artists the opportunity to add more publishable items or remove a selected publish item from the list. On Shotgun, upload versions are created for each published item from the tool and displayed in the overview section of the project. During a publish, logs are printed out to a dedicated window to show the status of a publish or any errors that may have occurred.

Although the applications has been tested and successfully used in small projects like “**Brave Player**”, it has not been battle tested in a full production. The application has also not been tested with all possible media formats that are supported by Shotgun. Although Shotgun

allows artist to publish workfile data about a current task, this tool does not support that functionality. These may be “chinks in its armor” but with moderated enhancement to the tool, these features can be supported.

5.2 Tractor Queue Submit

Tractor Queue Submit is an application that was developed for submitting batch render jobs from Maya to Tractor. Tractor is a network renderfarm manager that allows a set of computers to perform several tasks simultaneously. Tractor also lets an artist monitor the status of a task through a Web interface. The manager, also known as Tractor-Engine, is responsible for tracking, scheduling, and distributing job tasks. Renders are submitted to the Tractor-Engine as a queued job, through an “*Alfred job file*”. This job may consist of several tasks. Each task in the job is distributed to a computer, also known as a Tractor-Blade. After a task is completed, the Tractor-Blades report to the Tractor-Engine and wait for the next assignment. Tractor comes with a Job Authoring Python Application Programming Interface (API) [15] that allows artists and developers to create and customize job files submitted to a Tractor-Engine. Here is an example of the contents of a job file:

```
Job -title {Environment Job} -subtasks {Task -title {Environment Task} -cmds  
{RemoteCmd {{printenv}} -service {pixarRender}}}.
```

This command prints the environment variables that are set on the Tractor-Blade after the task is completed. Connecting Maya with Tractor required using the Job Authoring API to create custom task commands to run on the computers. To make the job submission easier for the artist, a graphical user interface was developed for artists. Since the render settings in Maya determine the results of a render, the interface fields are pre-populated when the application is launched from settings made in Maya. Figure 5.2 shows the Tractor submit application.

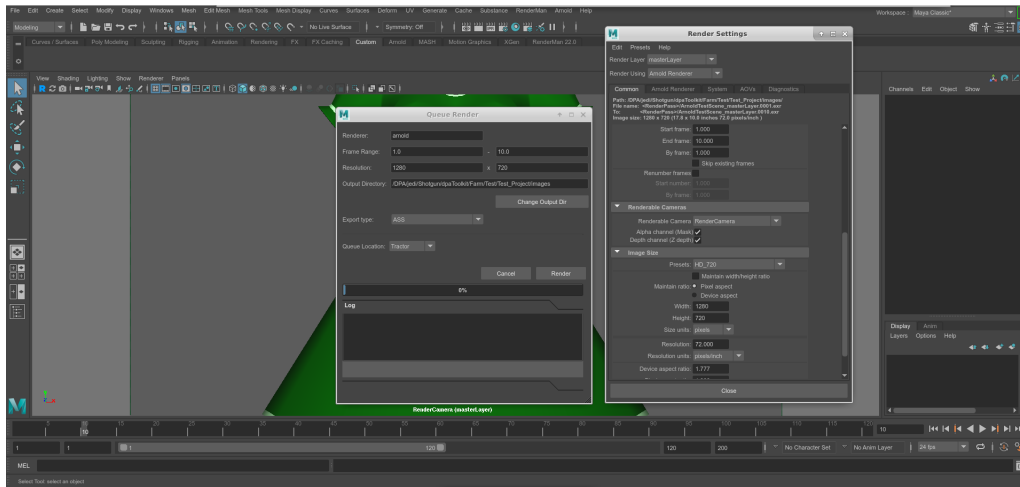


Figure 5.2: Tractor Render Queue Submission tool

In the image above, the submission tool is placed on the left of Maya's Render settings window to show the settings queried from the scene. The different fields on the graphical user interface(GUI) represent the important details about the render. The "Renderer" tab shows the current renderer used in the scene. This tab also determines the supported export type of the application. "Frame Range" is the number of animated key frames from the scene to render. This tab is non-editable and requires the artist to make changes in the render settings. If there are no animated key frames, this tab will show the current selected key in Maya's timeline as the render key frame. The "Resolution" tab shows the target resolution for the key frame image. "Output directory" shows the directory rendered images would be saved. This tab also allows the artists to change the final image location after the scene has been rendered. The last two options, "Export Type" and "Queue Location", determine how the final render is processed. Export type has several options determined by the renderer. Arnold exports either an Arnold Scene Source(ASS) file or a MayaBatch file. Renderman exports a Renderman Interface Bytestream(RIB) and a MayaBatch scene file. MayaBatch is a duplicate Maya scene export. Currently, Renderman and Arnold are the only supported renderers by the application and Tractor. When the artist submits a render, the application determines how to process the scene based on the export type and location. It then creates a Job file that is sent to the Tractor manager. This process is called "spooling a render job". After the job has been spooled, the status of the job can be monitored from a Web browser by searching the Domain Name Server(DNS) of the Tractor-Engine.

Tractor queue submit tool mimics how most batch commands are launched for renders in Maya. It has not been extended to support special commands such as denoising in most renderers like Renderman or Arnold. Denoising is a post process system used to clean up noise done after a frame is rendered. To support these type of features including Xgen, Mayabatch export type should be used as the rendering process.

5.3 Toolkit: Ubuntu Linux

Ubuntu is one of many different versions of the Linux operating system. Most motion picture studios use Linux in their pipelines because of its scalability and cost effectiveness [8]. Software like Autodesk Maya, Nuke, and Mari have support for Linux. The software are typically Redhat-based distributions like CentOS and Fedora. Shotgun Desktop is supported on CentOS and Fedora, not Ubuntu. Shotgun Desktop is the bridge that connects Shotgun and Toolkit projects on operating systems like Windows and MacOS. Before Shotgun Desktop, Toolkit used a Legacy commandline activation script to download configuration files that run Toolkit[16]. This activation script was later re-purposed into a user friendly interface now know as Shotgun Desktop. Although the activation script is a Legacy tool, its functionality is currently supported by the Shotgun team. During activation, a master configuration file is downloaded into a directory of the user's choosing. This directory contains a file that run commands to launch toolkit. This file, in combination with a custom Linux commandline script, allows an artist to launch a project on an Ubuntu operating system. Adding Ubuntu to the suite of operating systems supported by Toolkit, allows artist to use software not supported on Linux like Zbrush and Adobe Photoshop. This setup creates an effective pipeline for artist on different operating systems.

Although Toolkit on the back-end is mostly python, there are possible libraries that may not be available on the Ubuntu distribution and since Toolkit is not officially supported on Ubuntu, help from the Shotgun Toolkit support team is limited. This could hinder productions using tools on the Ubuntu operating system that error on launch.

Chapter 6

Conclusion

Pipelines account for a large part of the production budget and is fundamental in keeping projects organized, give artists easier opportunities to share work, and reduce bottlenecks during production. Clemson University's Digital Production Arts program has benefited greatly from its pipeline, DPA-pipe and Shotgun Toolkit. As story ideas become more complex and require more resources, pipelines need to evolve to suit the needs of the production. They also need to be efficient and user friendly by giving artists the chance to start and iterate over work easily and quickly. Less time spent in the details of the pipeline increases the productivity of an artist.

6.1 Analysis

DPA-pipe and Shotgun Toolkit overlap in functionality but differentiate on how those functionalities are presented to the artists. Below is a comparison of both pipelines; their similarities and differences.

6.1.1 Similarities

- **User Interface:** DPA-pipe and Toolkit both have simple straight forward user interfaces that allow artists to quickly and easily use its features. Although different in the user interface and approach presented to the user, they both perform similar task. An example is the publish and loader application. DPA-pipe and Toolkit both have a single click feature that scans the current work scene and parses deliverable types to be passed to the next artist. This gives

both the receiving and publishing artists easy and simplified export and loading functionality.

- **Application Support:** DPA-pipe and Toolkit both support industry standard artist tools like Nuke, Autodesk Maya, Houdini and Mari. This allows the artist to use the application and workflow they are most familiar, with the added bonus of easily being able to pass workfiles and publishes off to the next artist to continue work.

6.1.2 Differences

- **Multi-OS Support:** Unlike Toolkit, DPA-pipe current only supports the Ubuntu Linux. Although designed for Unix operating systems, it has not been fully test on other flavors of Linux or MacOS.
- **Project Setup:** Toolkit project setup is done through Shotgun. The project manager or director sets up the assets, shots, sequences, and task on Shotgun. Through a setup process on Shotgun Web and Shotgun Desktop, a technical director can create the folder structures and pipeline integration system for the project.

DPA-pipe uses a commandline approach to project setup. The pipeline expects the artist to know the structure of the pipeline and how files can be arrange to better suit the project before hand. Then the artist manually goes through a commandline process of creating the directory structures of the project. Tools and software integration are automatically added during the setup process.

DPA-pipe Shotgun publisher, gives artists the opportunity to submit results of completed tasks for review on Shotgun. Artist do not have to open a web browser and upload a rendered image to the web server. This automated process increases efficiency in the pipeline. The simple user friendly interface also allows first time users to quickly learn how to use the tool. Following the same simple user interface, the tractor queue submission tool allows artists to submit render jobs to a renderfarm with little to no advance knowledge of the farm.

Toolkit is a pipeline widely used in the industry. A multi operating system pipeline, gives artists and users the ability to use software that are not available on one system. This functionality makes a production easily scalable and portable. Shotgun and Toolkit integration also allow artists to quickly create projects and start working within a pipeline.

6.2 Future Implementations

- **Tractor on Palmetto Cluster:** Palmetto Cluster is a High Performance Computing(HPC) facility used for distributed computing[19] in research at Clemson University. Palmetto currently boasts a cluster of 2021 computers, totaling 23072 CPU cores. The Tractor Submission application can be configured to submit jobs to the Portable Batch Scheduler(PBS) on Palmetto. Due to modern network security protocols(2 factor authentication) applied to Palmetto and the current structure of running software on the HPC servers, a modern HPC software packaging approach should be used to run jobs on the cluster.

On Linux, the Secure Shell Filesystem(SSHFS) protocol can be used to load the project directory on Palmetto after authentication. This protocol allows file changes made on a local host computer reflect on the remote host and vise-versa. Using a singularity container with the rendering software installed, an instance of the rendering job can be started from within the container with the render job project bound as a system file path. A container is a standard unit of software, that packages up code and all its dependencies, so an application runs quickly and reliably from one computing environment to another [11].

A container will allow jobs rendering on the cluster appear local. No software would need to be directly installed on each computer. Using a container will also allow artist and technical artist to embed and maintain custom software and actions that can be ran on the cluster with minimal intervention from system administrators. Clustered computers are optimized to run a single task very quickly making it sometimes 10x faster than an artist's workstation. Running a render on several computers allows artists to get render jobs back in $((\text{number_of_tasks} / \text{number_of_nodes}) \times \text{time_per_frame})$ amount of time. See Appendix B for the container setup recipe.

- **File Cache System:** Caching is a system that allows data to be stored in memory or on disks which can later be accessed quickly. Adding a caching system to the publish pipeline, will allow artists to work from remote locations with good write speeds to files. When the artist is connected back into the pipeline, the cached file is uploaded to the right project location on the shared network and version up. This system will reduce the bottle neck in network connectivity to an off-location shared drive.

Appendices

Appendix A Statistics Questionnaire

- What type of Production have you worked on? Check all that apply
 - Animated Short Film
 - Animated Feature Film
 - Live Action Production
 - In Game Cinematic
 - Game Production
- Have you used a Production Pipeline before?
 - Yes
 - No
- Have you used the DPA pipeline before?
 - Yes
 - No
- Rate your skill level on a scale of 1 to 5
 - 1 2 3 4 5
- Have you used Shotgun Web Project Manager before?
 - Yes
 - No
- Rate your skill level on a scale of 1 to 5
 - 1 2 3 4 5
- Have you used Shotgun Toolkit before?
 - Yes
 - No
- Rate your skill level on a scale of 1 to 5

- 1 2 3 4 5
- What software do you use for Modeling? Check all that apply
 - Maya
 - Blender
 - 3DS Max
 - ZBrush
- What software do you use for Texturing? Check all that apply
 - Photoshop
 - Substance Painter and Designer
- What Renderer and software do you use for Look Development and Lighting? Check all that apply
 - Arnold
 - Renderman
 - Maya
 - Katana
 - Mental Ray
- What software do you use for animation? Check all that apply
 - Maya
 - Blender
 - Motion Builder
- What software do you use for Compositing? Check all that apply
 - Nuke
 - Fusion
 - Photoshop
 - After Effects

- Do you have a game development production experience?
 - Yes
 - No
- What game engine have you used for game productions?
 - Unreal Engine
 - Unity 3D
 - CryEngine
- What Operating System do you mostly work on?
 - Windows
 - MacOS
 - Linux
- Where do you mostly do your work?
 - All remote from home
 - All from On-Campus workstations
 - Campus workstations for big task but mostly work from home
 - Mostly on-campus but sometimes work from home when I have time
- Explain in a few words what you think a pipeline is
- What would you like to see in a pipeline?

Appendix B Palmetto Singularity Recipe

```
Bootstrap: docker

From: oraclelinux

# Lets install all the files we need in the base setup

%help

This is a palmetto container for DPA pipeline. Use the -app switch to run specific apps in the
container

%label

CREATOR

MAINTAINER

VERSION 1.0

COMPANY DIGITAL PRODUCTION ARTS CLEMSON UNIVERSITY

LICENSE


# set some persistent environment variables to use at run time e.g License info

%environment

ADSKFLEX_LICENSE_FILE=@license.server.clemson.edu

# Copy files into the container to be process later

%files

/tmp/singularity/configs/TrEnvHandler.py / # Tractor environment variables parser
/tmp/singularity/configs/pixar.license / # Pixar License file
/tmp/singularity/configs/shared.linux.envkeys / # Custom tractor envkeys
/tmp/singularity/configs/shared.macosx.envkeys /
/tmp/singularity/configs/shared.windows.envkeys /
/tmp/singularity/configs/License.env / # Maya License file
/tmp/singularity/configs/maya.lic /


# Ran at build time but after the OS is installed

%post

#####
```

```

# Software and custom system setup
#####
# Lets update our system files
    yum update -y
# Install some useful utilities
    yum install -y less which perl lshw csh tcsh vim vi nano sed
# change to the tmp directory
cd /tmp/singularity/Installers
    ##### Install Maya
#cd Maya
#rpm -ivh *.rpm
    # Install Maya dependencies
#yum install -y libXp libXmu libXpm libXi libGL libGLU libtiff
#libtiff-devel libpng libXinerama fontconfig fam libXrender
#libXcomposite libxslt libpng12 pulseaudio libXrandr tbb tbb-devel
#compat-libtiff3
    # Link some libraries for the license
#ln -s /opt/Autodesk/Adlm/R14/lib64/libadlmPIT.so /usr/lib64/libadlmPIT.so
#ln -s /opt/Autodesk/Adlm/R14/lib64/libadlmutil.so /usr/lib64/libadlmutil.so
    # Setup Maya licenses
#cp /License.env /usr/autodesk/maya2018/bin/ # Copy License file to Maya directory
#cp /maya.lic /var/flexlm
#/usr/autodesk/maya2018/bin/licensechooser /usr/autodesk/maya2018 network unlimited maya
#echo "export LD_LIBRARY_PATH=/opt/Autodesk/Adlm/R14/lib64" >>$SINGULARITY_ENVIRONMENT
    # Register the product to autodesk license server
#/usr/autodesk/maya2018/bin/adlmreg -i N product_key product_key 2018.0.0 serial_number
/var/opt/Autodesk/Adlm/Maya2018/MayaConfig.pit
    # Fix Maya permissions on folders
#chmod -R u+rx /var/opt/Autodesk/
#chmod -R g+rx /var/opt/Autodesk/
#chmod -R o+rx /var/opt/Autodesk/

```

```

#chmod -R u+rx /usr/local/share/macrovision/storage/
#chmod -R g+rx /usr/local/share/macrovision/storage/
#chmod -R o+rx /usr/local/share/macrovision/storage/

# I know we have more rpm's so lets install them. (Tractor, RPS, RFM. RFK)

#cd ../

#rpm -ivh *.rpm

##### Setup Tractor
ln /opt/pixar/Tractor-2.2/lib/SystemServices/systemd/tractor-engine /etc/default/
ln /opt/pixar/Tractor-2.2/lib/SystemServices/systemd/tractor-engine.service /etc/systemd/system
ln -s /opt/pixar/Tractor-2.2/lib/SystemServices/tractor-engine /etc/init.d/
ln -s /opt/pixar/Tractor-2.2/lib/SystemServices/tractor-engine
/etc/systemd/system/multi-user.target.wants/

##### Install Arnold
# Install Arnold for Maya, might have to manually do this because of license agreeemnt, first make
it executable
# chmod +x MtoA-3.0.1.1-linux-2018.run # execute it
#./MtoA-3.0.1.1-linux-2018.run

##### Install Katana 3.0
# unpack the file into a different directory, might have to install manually because of license agree-
ment
# mkdir katana && tar -xvzf Katana3.0v3-linux-x86-release-64.tgz -C katana
# cd katana
# ./install.sh

##### Install Arnold for Katana
# mkdir ktoa && tar -xvzf Ktoa-1.0.3-linux.tgz -C ktoa
# cd ktoa

##### Setup environment variables
echo "export=ADSKFLEX_LICENSE_FILE=license.server.clemson.ed" >>$SINGULARITY_ENVIRONMENT
#####

```

```

# Setup specific to pipeline

#####

# Lets setup Tractor to match our custom Setup
mv -f /pixar.license /opt/pixar/pixar.license
mv -f /TrEnvHandler.py /opt/pixar/Tractor-2.2/lib/python2.7/site-packages/tractor/apps/blade/TrEnvHandler.py
mv -f /shared.linux.envkeys /opt/pixar/Tractor-2.2/config/shared.linux.envkeys
mv -f /shared.macosx.envkeys /opt/pixar/Tractor-2.2/config/shared.macosx.envkeys
mv -f /shared.windows.envkeys /opt/pixar/Tractor-2.2/config/shared.windows.envkeys

#####

# App specific setup on run or exec

#####

##### Tractor

##### ENGINE

%apprun Tractor-engine
echo "Sourcing config file"
#Source Config
#MAX_CONCURRENCY=100
echo "Running modified configurations...."
echo ""
echo ""

#sed -i 's|"MaxConcurrentDispatch" : 0,|"MaxConcurrentDispatch" : ${MAX_CONCURRENCY},|'
/opt/pixar/Tractor2.2/config/tractor.config
echo"ConfigurationComplete!Startingtractorengine...."
echo""

servicetractor --enginestart

%appenvTractor --engine
%applabelsTractor --engine
TRACTOR Pixar TRACTOR Engine
VERSION 2.2
%apphelpTractor --engine

```

TractorEngineforrenderQueuemanager

```
##### BLADE
%apprun Tractor-blade
echo "Tractor blade is running"
%appenv Tractor-blade
%applabels Tractor-blade
VENDOR Pixar TRACTOR Blade
VERSION 2.2
%apphelp Tractor-blade
Tractor Blade for render Queue manager
```

```
##### Katana
%apprun Katana
echo "Katana is Installed"
%appenv Katana
%applabels Katana
VENDOR THE FOUNDRY
VERSION 3.0
%apphelp Katana
Tractor Engine for render Queue manager
```

```
##### Maya
#%apprun Maya
# echo "Maya is starting up.."
# printenv
# /usr/autodesk/maya2018/bin/maya2018 -prompt
#%appenv Maya
MAYA_DISABLE_CIP=1
#%applabels Maya
# VENDOR Autodesk
```

```
# VERSION 2018
#%apphelp Maya
# Maya 2018 installed in the default location
```

```
##### Job
%apprun Job
echo "Spooling to Tractor"
%appenv Job
MAYA_DISABLE_CIP=1
%applabels Job
JOB Tractor Batch Job
%apphelp Job
This an app to spool an alf file to tractor
```

Bibliography

- [1] Pipeline/ptask basics, 2015.
- [2] Michael Balog Michael Clausen Gavin Moran Brian J. Pohl, Andrew Harris and Ryan Brucks. Fortnite: Supercharging cg animation pipelines with game engine technology, 2017.
- [3] Josh Tomlinson Nico Van den Bosch Wil Whaley Chris Johnson, Josef Tobiska. Rhythm & hues - a framework for global visual effects production pipelines - siggraph 2014.
- [4] Josh Tomlinson Nico Van den Bosch Wil Whaley Chris Johnson, Josef Tobiska. A framework for global visual effects production pipelines, 2014.
- [5] Timothy Curtis. Efficient control of assets in a modern production pipeline. Master’s thesis, Clemson University, 2014.
- [6] Nicole DeMichelis. Vfx librarianship: designing a global asset library for a visual effects studio, 2016.
- [7] Bill Desowitz. ‘spider-man: Into the spider-verse’: Breaking the rules of animation, 2018.
- [8] Aaron Estrada. The history of linux in vfx and animation, 2017.
- [9] Ian Failes. “if it’s not broke, break it: Sony imageworks’ renegade approach to ‘spider-man: Into the spider-verse’”, 2018.
- [10] Sony Pictures Imageworks. ‘spider-man: Into the spider-verse’, 2018.
- [11] Docker Inc. What is a container, 2019.
- [12] Don Parker Josh Tomlinson, Manne Ohrstrom. The shotgun pipeline toolkit: Productizing and democratizing production pipelines, 2017.
- [13] Alasdair Coull J.P. Lewis, Cristian S. Calude. Can we solve the pipeline problem?, 2014.
- [14] Erik Smitt Mahyar Abousaeed and Leif Pedersen. Incredible cinematography, 2019.
- [15] Pixar. Job author python api, 2018.
- [16] Autodesk Shotgun. Activating toolkit via the command line, 2018.
- [17] Autodesk Shotgun. Shotgun, 2019.
- [18] Autodesk Shotgun. Shotgun python api3, 2019.
- [19] Clemson University. Clemson university palmetto cluster, 2019.
- [20] Wikipedia. Project management, 2019.

- [21] Wikipedia. Shot(filmmaking), 2019.
- [22] YourDictionary. Renderer, 2019.
- [23] Jordan Zakarin. Spider-man: Into the spider-verse required inventing a new kind of animation technology, 2018.