FROM PRODUCTION TO EDUCATION: AN ANALYSIS OF PIPELINE REQUIREMENTS AND PRACTICES

A Thesis

by

BRANDON LEE JARRATT

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Frederic I. Parke
Committee Members,	Ann McNamara
	Tracy Hammond
Department Head,	Tim McLaughlin

May 2013

Major Subject: Visualization

ABSTRACT

Animation, visual effects, and video game studios have to manage complex and highly iterative productions. The processes, tools, and data flow that carry a production from initial idea to finished state is called a 'pipeline.' Students in academic programs, even ones focused on educating for digital production, often do not have a well-defined pipeline and spend unnecessary time on technical details rather than creative work. Through interviews with industry professionals, analysis of published works on pipeline and digital production, and study of current academic pipelines, this thesis presents general principles for pipelines as well as suggestions for applying these principles in academic environments. Implementing these suggestions could provide a foundation for a robust academic pipeline that lets students spend more time creating and collaborating and prepares them for employment in the digital production industry.

ACKNOWLEDGEMENTS

Thank you to all the industry professionals who agreed to be interviewed. Your time and generous discussion are the cornerstone of this thesis.

I would like to thank Dr. Parke for his guidance and patience during the writing and approval process, and my other committee members for their feedback and suggestions about the goals of my research.

Dr. Hank Driskill was instrumental in putting me in touch with other industry professionals for interviews. My time working with him was the inspiration for this thesis.

Thank you to my family for letting me shut myself away to write while you were planning a wedding. Neither my wedding nor my thesis would have happened without you.

Finally, thanks and love to my wife Savannah for her support and patience during this long process.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The goal of this thesis is to study pipeline practices at digital production facilities for animation, games, and visual effects to determine some general principles about pipelines. The research is conducted through interviews, literature search, and study of existing academic pipelines, with the goal of finding pipeline principles that can be applied to academic practice. I will offer suggestions as to how these ideas might be used in an academic setting.

I.1. Introduction to the Concept of a Pipeline

A pipeline, understood in the context of computer animation, is a set of processes and tools that transport data through the various stages of production, to the final product. Every major visual effects (VFX) or feature animation production studio utilizes a pipeline to manage data and workflows, enhance communication, increase efficiency, and aid troubleshooting when things go wrong. The end-to-end combination of workflows, with output from one stage of production becoming input for the next, is the definition of a pipeline.

The notion of a pipeline as a linear structure, carrying items in sequence from one end to the other, is not adequate. Digital production processes are complex and highly iterative, requiring constant communication, extensive record keeping, version control, and the ability to handle multiple feedback loops.

A pipeline includes everyone involved in the production, from executive producer and director down to the artists painting textures and creating 3D models. Production personnel work together to carry the project from the idea in the director's mind into

a complex collection of assets and shots that tell the intended story. An asset is simply a single unit of production, such as a 3D character model, set or environment, texture, or a virtual light or camera. A shot, following the convention of live-action films, is an action sequence that tells part of the story.

In feature animation, the process begins by creating a story script. The story script is the basis for the project. It defines the dialog, storyboards, and overall design of the film. The process of finalizing the story script can take years. Once the story script is approved for production, storyboards can be made to visualize the action of the script [23]. Designs for the film's characters, props, sets, effects, and lighting also begin at this time.

Artists and workers with specialized skills, such as 3D modelers, programmers, or character riggers, are grouped into departments, based on their specialization, to handle specific stages of production. Each department has its own set of tasks that help the production move along. For example, the modeling department will take character designs from the art department and then create 3D models from those designs. These models are passed to the rigging department, where they are given skeletons and control systems for use by the animators. At each stage of production, these assets are reviewed, tested, critiqued as needed, and may be sent back to the artist with notes for revision.

Digital production is an iterative process. Several rounds of revision may be required before an asset is approved and sent on to the next department downstream. A department is 'downstream' when its work is dependent on receiving output from a different department in the pipeline. For example, the rigging department is downstream from the modeling department. The riggers need 3D character models for which to build skeletons and control systems.

Eventually assets are used to populate shot files. The content and action of

each shot are dictated by the film's storyboards. Assets, such as rigged characters, are staged, choreographed and animated for a virtual camera, which provides the viewpoint for the final image rendered by the computer.

Shots go through a sub-pipeline of their own, starting in layout (where the cameras and movement are blocked out), moving on through animation (where character performances are created), lighting, and effects (usually non-character animation, such as an explosion or waterfall). After the final images are rendered by the computer they are passed to the editorial department where music and dialog are added and synchronized to finish the film.

Feedback loops are common in digital productions so the pipeline should be able to handle them accordingly. Each asset will usually be archived and catalogued each time it is changed, building a useful revision history that charts the asset's progress through the production. Previous versions can be restored if the asset is accidentally deleted, causes errors for a downstream department, or if the director wishes to go back to an older version for creative reasons.

Before production begins a set of decisions must be made to address project needs. Directory structures, naming conventions, revision control methods, and the general path of data through the pipeline must be defined. Logical and predictable directories for assets and shot files should be established.

Channels of communication should be identified. To whom should an artist turn when he has a question? How do notes and suggestions get passed between departments? Who decides when an asset is ready for the final shot? How should an artist or programmer track and request bug fixes for tools? These questions are the basis upon which pipeline tools and workflows are built.

Once the workflow is determined, tools must be created to implement it. A recurring statement among industry professionals is that the best pipeline is the one

you don't think about. It is far more efficient for an artist to type a single command (or even better, to click a single button) that executes a set of tasks behind the scenes than for that same artist to manually enter a series of complex commands to achieve the same end result. The goal is to ease the burden on the artist, making it easy for him to create assets with limited worry about technical details like correct file naming or the capability to retrieve an earlier version of his work. Examples of some of these tools are discussed in section II.2.2.

Taking the technical details of file storage, naming, and revision control out of the artist's hands also decreases chances of making a mistake and causing problems downstream. For example, in a system where artists are able to name their own files, an artist might save a model file as 'Main Character Model.' The rigging department uses a tool that imports files matching the format 'model_mainChar.' Since the model file is not named correctly, the rigging department will be unable to import it until the modeler renames his file. Confining artists to a narrowly defined set of steps can mitigate problems like these.

Many pipeline tools are designed to automate common tasks. For example, a studio might use a program that runs at night to identify any assets that have changed since the previous night. Any shots containing these changed assets are then automatically re-rendered so that updated versions of the shots are available for review the next morning. Other tools might enhance the file saving operation by automatically generating text notes that are sent to artists in downstream departments.

Some workers need to be given special responsibility to ensure that these tools function correctly. These personnel need to possess a general knowledge of the artist's tools as well as programming skill and technical acumen. They are sometimes assigned to a department as technical directors, or TDs. TDs assist artists with software and workflow problems to keep production moving.

Most studios also have 'front-end' tools available for the artists. These are utilities that provide a user-friendly way for artists to interact with underlying pipeline systems. These tools might include navigation shortcuts (single-word commands that select specific directory areas), scene builder menus (a graphical user interface that allows users to choose assets for a shot), backup or file archiving commands, interfaces that import data from upstream departments, and any other function that addresses needs of the production.

Clear communication is essential. Communication must occur at all stages of the production process, between all departments. Production leadership has to figure out how to make the film using the financial and technical resources available. The director has to communicate his vision to the artists. The artists must communicate with each other and with the director to refine their work. Department supervisors must communicate with production management to ensure that work is getting done on time (and on budget). Artists must alert TDs to problems they encounter with assets and software. TDs must articulate the needs of artists to the software engineers who build pipeline tools. The software tools have to be able to communicate with each other and pass data along, sometimes between formats [16]. Good communication between all areas of production helps ensure that each asset and shot progresses smoothly through the pipeline.

This thesis is organized as follows: Chapter II summarizes related work. Chapter III discusses the methodology and process used for collecting data about industry pipelines. Chapter IV summarizes the background and responses of each interview participant in the research. Chapter V presents and discusses the results. Chapter VI concludes and discusses future work.

I.2. Motivation

Prior to 2011, students in the Texas A&M Visualization Laboratory lacked a defined pipeline of supporting tools to help them create and organize their projects. Students working on short animations created their own workflow and handled their own file directories [2]. Pipelines had been established for specific projects such as the annual summer industry course where professionals from studios like DreamWorks, Disney, and Pixar guided teams of students in the creation of short animated films. Project teams of four to six people used these pipelines to create, catalog, render, and store their thirty-second animations. But these tools and processes were generally not re-used after each project concluded.

Following the completion of the 2011 summer course, the graduate assistants to the Lab's system administrator (the developers) were tasked with building a general, lab-wide project pipeline. Both assistants had each established pipelines in previous summer courses. These pipelines provided directory structures and utilities for automating some tasks. The developers used their experiences to combine the best elements of each pipeline and provide new workflow and directory structures for student projects.

Several tools, in the form of command-line utilities, were provided to enable users to create and organize assets and move them into shot files in an informed manner. The primary form of interaction between the students and the pipeline was the Linux terminal command window. They could type simple commands to quickly perform useful tasks and avoid tedious ones. Terminal commands were chosen primarily for speed of user entry and execution, and to compensate for developer inexperience with building GUI tools.

This pipeline provides a basic structure, allowing students reasonable freedom

to manage assets and shots as they please. Fig. 1 illustrates the directory structures and data locations in the Lab pipeline. Fig. 2 shows an overview of the data flow in the pipeline.

At the start of the fall 2011 semester, this pipeline was introduced and recommended to all students as robust mechanism for managing projects. Students were presented with this basic workflow model. Tools to aid and automate parts of this process were demonstrated and discussed in presentations and Q&A sessions conducted by the developers. However, many students did not adopt the pipeline, despite its relative simplicity and useful utilities. As a result, many problems with projects occurred such as broken references, misplaced textures, use of incorrect file versions, incompatible path names, and so on. These could have been avoided by using the provided pipeline environment and tools.

It is unclear whether students found the pipeline overly complicated, the workflow creatively restricting, or were simply used to doing most of their work through GUIs rather than typing commands into a text-based terminal window. Some investigation was needed to determine how the pipeline could be improved to better suit the needs of students working on projects in an academic setting.

This thesis aims to compare and contrast industry pipeline needs with those of an academic facility, where the scale of production and consequences of user behavior can be very different. For example, most student projects are only accessed and maintained by one student, whereas industry projects need to be robust, support hundreds of users, and usually require sophisticated version control schemes.

Fig. 1. Directory Structure of the Visualization Laboratory Pipeline.

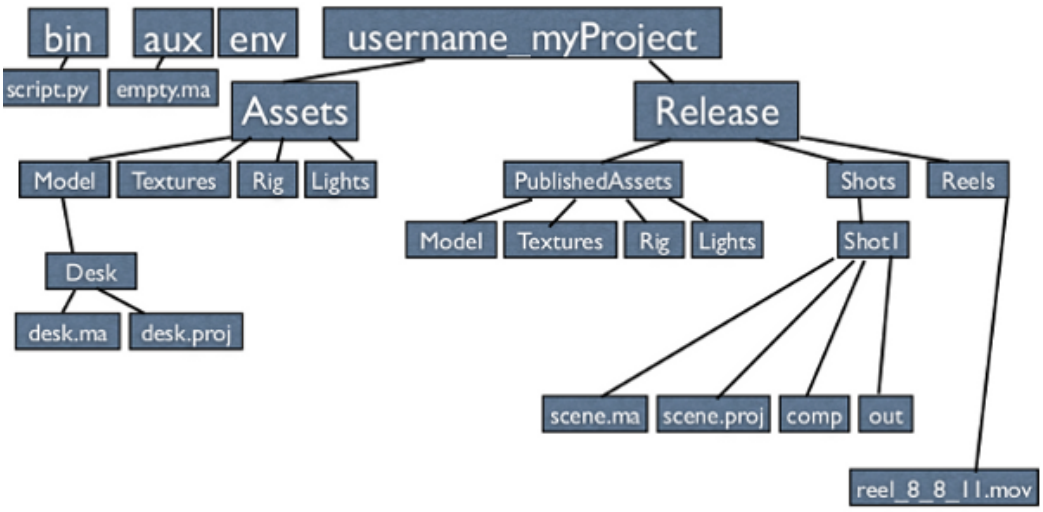
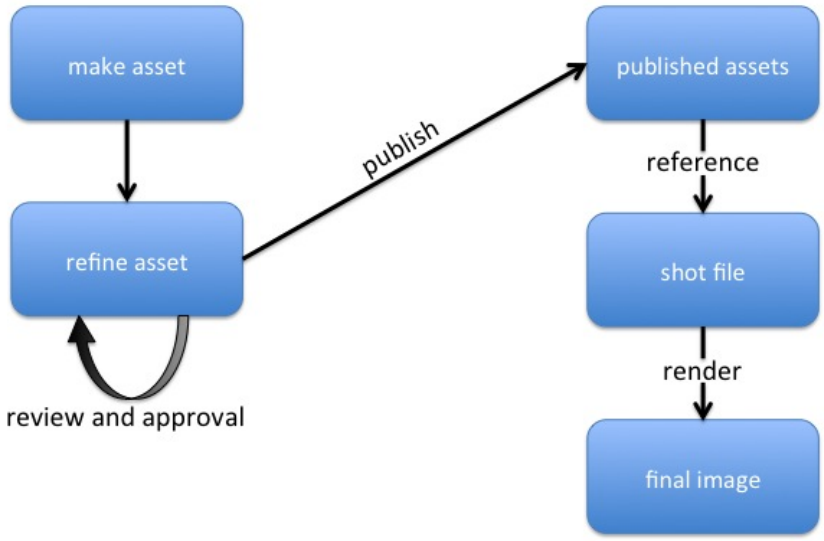


Fig. 2. Basic Data Flow in the Visualization Laboratory Pipeline.



CHAPTER II

RELATED WORK

This chapter discusses previous work that applies to this thesis. Section II.1 discusses the definition of a pipeline as determined by Bettis [3]. Section II.2 discusses the component parts of a digital production and methods of communication between them. Section II.3 studies existing academic programs focused on digital production. Finally, Section II.4 examines an open-source specification for production pipelines that is used in both education and industry [21].

II.1. Defining a Pipeline

This section examines an existing conceptual definition of a pipeline and how it relates to industry practice.

II.1.1. A Conceptual Definition

Existing discussions of academic pipelines are scant. Bettis' master's thesis from 2005 [3], which sought to provide a functional high-level definition of a pipeline, is the only published work this researcher could locate on the topic. Bettis explored the history of computer animation and compared the traditional animation process with the modern digital process. He performed a case study of a single animation studio, interviewing several employees to find emergent themes related to pipelines. He used these themes to form a conceptual definition of a pipeline.

Bettis' definition splits the pipeline into three layers: personnel arrangement, implementation and managing complexity, and optimization of computer systems. The first layer concerns demarcating groups of artists, assigning responsibility, and

defining relationships between these groups so that the best working process can be achieved. The second layer involves the implementation of the first layer into computer hardware and software, to automate and aid the artist's workflow and communication. The third layer is a refinement of the second layer that seeks to optimize its ability to be useful to the production. Examples of this include adding more, faster processing capability or improving an in-house software tool to more efficiently perform a necessary task.

Bettis' definition is limited in that it was based on the study of a single studio. However, it still provides a useful conceptual framework.

II.1.2. Definition Applied to Industry

In a series of blog posts written in 2009 [1], CG supervisor Isa Alsup deconstructed the pipeline based on his experience working in visual effects. Alsup established his own breakdown of the main pipeline into discrete related sub-pipelines - the *production* pipeline, *materials* or *data* pipeline, and the *approval* or *metadata* pipeline. The *production* pipeline is the assembly line collection of specialized skills that perform tasks. The *materials* pipeline manages the flow of data and assets through the project. The *approval* pipeline comprises the direction and instruction about a particular asset or shot, which includes revision history and capacity for looped feedback.

Alsup goes on in further posts to cite and discuss Bettis' research about pipelines and apply those concepts to several scenarios that might occur in production. He makes the distinction between the *technology* used for the production and the pipeline itself. The tools in the pipeline are just that - tools used by the artists as part of a workflow to accomplish tasks to be handed to the next department down the line. He restates Bettis' three-layer definition as three dimensions - personnel, tools, and procedures. He uses Bettis' work to support the points he makes in his articles.

Most importantly, he takes Bettis' conceptual definition and turns it into a functional description of CG pipelines, broken down into seven main points. According to Alsup, a pipeline:

1. belongs to one of three classes: task, data, or metadata.
2. is comprised of three dimensions: personnel, tools, and procedure.
3. utilizes technology but is not the technology.
4. divides a workflow into separate meaningful tasks for to two or more persons.
5. task divisions are determined by specialization across the three dimensions
6. structure is dicated by the functional mission, resources, and company culture.
7. is malleable.

II.2. Facets of Production

This section breaks down the three parts of Bettis' conceptual pipeline definition: personnel management, tools, and procedures.

II.2.1. Managing a Production

In their book *Producing Animation* [30], Catherine Winder and Zahra Dowlatabadi define and discuss the role and responsibilities of a producer in CG and 2D animation projects. Production planning, scheduling, budgeting, tracking, and communication, from beginning to end, are covered for large and small projects alike.

While this book is written from the perspective of the producer, it provided useful insight into the top-level decisions mentioned by some of the interview participants. For example, the producer must work with studio leadership to assemble

talent, identify the scale of the project, and assign roles and labor. These decisions all shape the production pipeline.

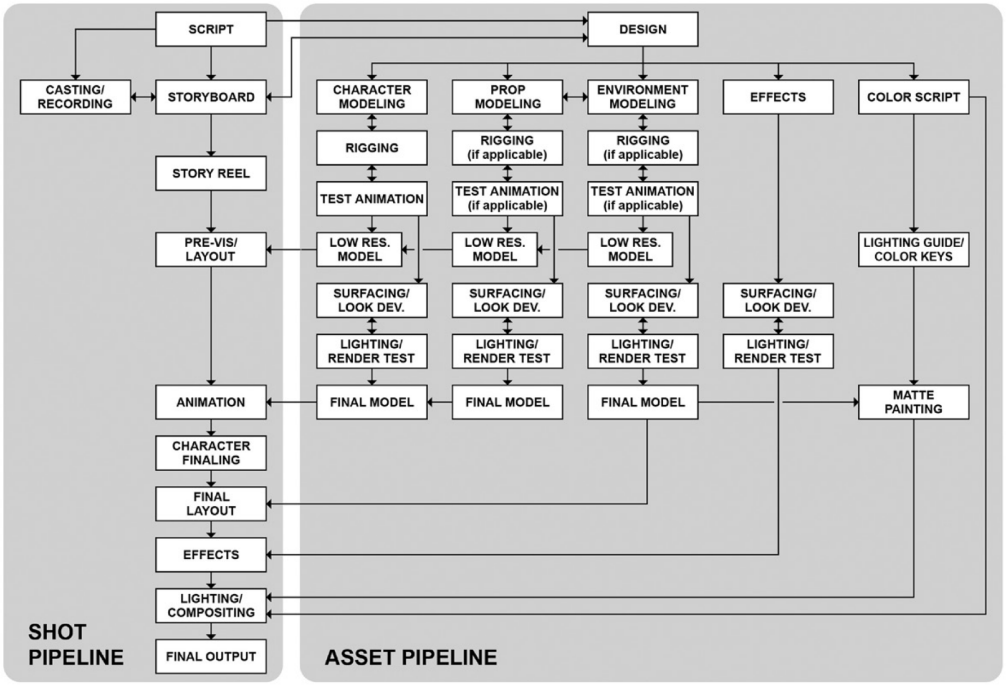
Production management is a critical component of any pipeline. The producer is responsible for establishing the leadership structure of the project and organizing the reviews and approvals process. Producers work with department supervisors to divide work among the artists to schedule a way that will get the project done on budget. The producer is also responsible for communicating the creative needs of the project to the visual effects supervisor who determines what technology and tools are needed to actually achieve the director's vision.

The director is the key creative decisionmaker and storyteller for the project. He or she is responsible for communicating the project vision effectively to the artists and the producers. The director must keep both the creative and financial needs of the project in mind when making decisions and assigning roles. Once production begins, the director will hand out assignments to artists, review their work, and give them feedback until the final look of the project is achieved. Winder and Dowlatabadi also provide an overview of the production pipeline and the creative departments involved in the project. Their illustration of the production process is shown in Fig. 3.

The details of creating and managing digital productions are covered in other sources as well. Chuang's presentation on building and scaling digital studios [7] touches on many of the same points as Winder and Dowlatabadi's book, but focuses more on the way that projects and studios grow over time as they face new challenges. Chuang classifies productions into one of three growth models - centralized, semi-distributed, and fully-distributed.

The centralized model, rooted in a single physical location, is the 'traditional' model that most people associate with a production studio. Here, the production team can take advantage of close proximity to communicate directly. Software and

Fig. 3. Typical Parallelized Production Pipeline. [30] ©2011 Catherine Winder and Zahra Dowlatabadi. Originally Published by Elsevier Inc. All rights reserved.



other pipeline tools are very similar across the entire facility, even if multiple projects are in production. Managing this production model is a matter of plotting personnel, dependencies, and time in a way that will get the work done.

The 'semi-distributed' model is becoming more prevalent, with many studios opening smaller satellite facilities in distant locations or subcontracting work to other studios. Semi-distributed studios employ a centralized model *at each location*, making the work, growth, and sometimes tools of each facility independent of the others. However, procedures must be defined to allow the transfer of data between locations.

In his talk "Keeping Your Money On The Screen & Off The Floor" [13], Kevin Geiger lays out good CG production principles and practices, as well as common pitfalls that lead to wasted resources. Geiger focuses on the human factors in a

production, such as effective managerial leadership, interpersonal communication, and usability of the pipeline for artists. He compares serial, or linear, pipeline structues with parallel or repository-based pipelines to show the benefits of non-linear pipelines.

As Geiger mentions, *people* are the most expensive and most valuable asset of any production. Finding the right balance between pressure, engagement, and productivity is key for getting through a project successfully. One key aspect of getting the most out of artists is the critique and feedback process. Evan Hirsch discusses methods for delivering creative feedback in his SIGGRAPH course notes from 2012 [15]. Hirsch states that clear, constructive feedback is crucial to helping artists feel engaged and empowered in their work. Vague instructions are hard to act upon, and overly negative or insensitive critique can even be counter-productive for artists.

Methods of production management have even been subject to academic research. Naugle's master's thesis from 2011 explores the application of Building Information Management (BIM) principles to animation production [19]. BIM was originally developed to increase efficiency in architecture projects. Naugle's research proposed a work management system for animation based on these principles.

II.2.2. Workflow, Tools, and Creative Solutions

The aspect of production most typically perceived as 'the pipeline' is the toolset employed to create and manage assets and shots. As discussed in earlier sections, the tools are only a *component part* of production, but nonetheless a crucial part that can greatly affect capability and efficiency.

From the earliest days of animation, technology and tools have been implemented to enhance creativity and break new artistic ground. The multiplane camera used at Walt Disney Studios beginning in the 1930s allowed for more depth in the background and more believable camera movement through animated environments [27]. Break-

ing the camera view into individually movable layers is a technique that influenced subsequent animation and compositing for visual effects.

Computer systems to aid and create animation began appearing in 1970s. Catmull presented an overview of these sytems and pointed out the problems and challenges of computer-assisted animation at that time [6]. Computer animation systems then were used primarily for scanning, coloring, and creating inbetweens for 2D animated character sequences. One such system is the Computer Animation Production System (CAPS) [26] created for Disney in the late 1980s. Disney's *The Rescuers Down Under* (1990), made using CAPS, was the first completely digitally produced feature film.

CAPS and other computer tools were introduced as a way to improve existing production methods, or to replace analog versions entirely. With CAPS, for example, the expensive and time-consuming process of inking and painting cels was replaced with a much faster and more flexible computer-based flood-fill coloring system. In modern digital productions, computer-based tools may be incorporated into the pipeline for similar reasons. At Mainframe Entertainment, for example, physical note-passing was used as a method of transferring shot responsibility between artists until the studio implemented an asset manager software tool [25].

Sometimes tools and workflows must be created to address new production needs not previously encountered by the studio. Before working on *Rango*, Industrial Light + Magic (ILM) had never done a feature-length animated film. They had to implement a new materials library and a lighting workflow that could be re-used for many shots in order to handle the number and complexity of the film's assets [10].

Similarly, The Moving Picture Company (MPC) had to change the way they dealt with assets in order to handle a large number of shots for *The Chronicles of Narnia: Prince Caspian* [5], and again for *Prince of Persia* [18]. Rather than rely on a

hierarchical structure of assets based on shot number, name, and version, MPC moved to a system based on the relationships and dependencies between assets. Instead of treating assets such as models, rigs, and animation caches separately, they were grouped into 'packages' associated with a particular character, scene, or object. This additional relationship allowed changes on one asset in the package to automatically affect other assets. In addition, these packages were categorized into different types that could be simultaneously worked on by different departments. An approvals system was set up to control the merging of work from different departments into asset packages.

Assets and work will sometimes need to be created and modified in different software applications, depending on the needs of the shot and the preference of the artist. However, most 3D software tools (such as Maya or Houdini) have limited support for exporting entire scenes with animation, lighting, and surface elements intact. To allow their artists and technical directors to construct each portion of a scene in their preferred tool, Digital Domain devised a system to import and export scenes while handling different geometry and scene hierarchies [16].

Some projects have very specific needs driven by the story or chosen art style. For example, the 2010 film *Legend of the Guardians: The Owls of GaHoole* featured a cast of emotive owl characters that all required feathers. These highly realistic feathers needed to be adjustable by the artists to create specific shapes and poses in each shot. This required them to be modeled, articulated, and simulated. Animal Logic, the creators of the film, already had a proprietary procedural animation and simulation engine called ALF in their pipeline. But, it did not have the capability to produce the feathers they needed for this film. Since no commercially available package existed that could meet their needs, Animal Logic extended ALF with a procedural feathering tool called Quill [14]. Quill enabled modeling, surfacing, animation, effects

and rendering of feathers. As an addition to Animal Logics production pipeline, it allowed the artists to adhere very closely to the concept art for the characters and maintain a high level of realism.

Another example of tools driven by art and story requirements are the hand-held camera and ocean wave systems created by Sony Pictures Imageworks for their 2007 film *Surf's Up* [4]. Since the film was styled as a surfing documentary similar to *The Endless Summer* or *Step Into Liquid*, they had to find a way to mimic the hand-held camerawork and middle-budget production values of those films. To achieve this look, they built a new live-action camera system for the film, dubbed the HandyCam, which allowed a live action camera to be used to shoot an animated scene. The camera operator would operate the physical camera and the capture system would give instant feedback of the performance in the form of a smooth shaded virtual 3D world fed back into the eyepiece of the camera.

Because of the film's subject matter, the other major element needed for production was the creation, rigging, animation, and rendering of realistic ocean waves. Imageworks had many challenges: create a visually realistic wave, allow for a high level of artistic control, handle the unique interdependence and overlap between animation, effects and layout, and design a production pipeline to produce the wave as efficiently as possible.

Finally, with the recent increase in "3D" or stereographic films, studios and visual effects houses have had to incorporate new techniques and tools into their pipelines to support stereo imaging methods. Damien Fagnou of MPC presents a brief overview of these methods and some of the technology used to support them [11]. The first and most expensive option, native stereo, involves shooting a scene with two cameras that are aligned and synchronized to capture the action from each 'eye' viewpoint. This method mimics the way human eyes view 3D images, but is prone to misalignment and

distortion and usually requires adjustment by a software program in post-production. The second method, post-conversion, is utilized when a film shot in standard 2D is also given a 3D release. It requires rotoscoping a large portion of the image to isolate the parts that need to be at different depths. Post-conversion is very time-consuming and generally does not yield the same image quality as native stereo.

II.3. Preparing Digital Artists for Industry

Multi-disciplinary academic programs geared toward digital production have existed since the 1980s. Texas A&M's Visualization program, for example, has sent many graduates on to careers in animation, visual effects, and related fields. With the increasing prevalence of computer graphics in Hollywood films in the early 1990s, several other programs, such as the University of Southern California's Computer Animation Laboratory [28] were founded to prepare students for work in these new areas of production.

The common thread among these programs is a focus on *collaboration* and *interdisciplinary work*. Courses taught at Purdue [9] [29], The College of New Jersey [31], Bournemouth University [8], and Texas A&M University [17] allow students to work together, sometimes partnered with industry or other university departments, to complete animation, video game, or visualization projects. This thesis aims to improve the pipelines in these kinds of programs by applying industry practices to their design and implementation.

II.4. An Open-Source Pipeline Specification

Production studios are not the only source of information about pipeline practice. Some efforts have also been made to develop open-source project pipelines. One such

effort is openPipeline [21]. Developed at Pratt Institute's Digital Arts Department, openPipeline offers both a specification for pipelines and a software implementation. The specification defines elements of a production, overall directory structure, general workflow, and rules. The software provides automatic directory structures, file naming conventions, and revision control as part of a MEL-based Maya plug-in. openPipeline is intended to emulate the functionality of most studio pipelines and to support a similar level of complexity. What makes openPipeline unique is its release and licensing as open-source software. openPipeline encourages distribution, modification, and contribution to its source code and specification. Students have used openPipeline to manage and produce short films. Several digital production studios have even adopted the software and specification for use in their work [20].

CHAPTER III

METHODOLOGY AND PROCESS

This thesis work has three components. The first is an investigation of the structures and functionality of different industry production pipelines. Second, these practices are analyzed to determine general principles that may exist. These principles and industry practices are compared with existing academic project pipelines. Finally, guidelines for academic pipelines are suggested.

III.1. Interviews

The primary source of information for this research is a series of seven interviews with industry professionals from different production studios. By gathering a large cross-section, this research aims to draw conclusions about the common elements between pipelines in feature animation, television, visual effects, and games.

Participants were chosen based on their job titles and relevant experience with production pipelines. In many cases participants were referred to the researcher by previously-established industry contacts. Each participant was contacted by email with an overview of the research goals and the primary interview questions (listed below).

After asserting their initial willingness to participate, each interviewee submitted their written permission to be interviewed and audio recorded, in keeping with the human subjects research protocol approved by the Institutional Review Board (IRB). In some cases, approval from the studio legal department was required before participants were allowed to proceed. In all cases, interviewees and their employers were given the right to review the interview transcript and redact any information

they deemed inaccurate or proprietary.

Each interview lasted between thirty and sixty-five minutes, depending on the complexity and length of the participant's answers. After talking briefly about their background, experience, and position at their studio, the participants were all asked the same set of basic questions, along with any relevant follow-up questions the researcher felt were appropriate in the course of the interview. The questions posed by the researcher were as follows:

- How would you define a pipeline?
- What characterizes a *good* pipeline?
- What kinds of feedback loops and/or communication tools make a pipeline effective?
- How important are version control and/or change-tracking methods?
- What kind of flexibility do you need in a pipeline to handle different project requirements?
- Who decides what's part of a pipeline? Are these consensus decisions, or dictated by small groups of people?
- How have your pipeline systems evolved since you started? Are there things you'd like to improve about it?

Interview data was coded to preserve confidentiality and to prevent preconceived notions about the studios from affecting the analysis. Participant names, studios, and film titles were replaced according to the IRB protocol. These changes are reflected in the full interview transcripts presented in the appendices.

Upon completeing the interviews, a full text transcript was created from each audio file to be analyzed for patterns in the responses. Some questions that help to frame this analysis:

- Who are the stakeholders in a pipeline?
- What factors influence the decisions that are made about pipelines?
- Who makes the decisions about pipelines?
- Are there identifiable patterns in basic structures across different studios?
- What common threads can be drawn between large, department-based studios and smaller visual effects shops?
- What are the types of tools that artists need to most easily move production from one stage to the next?
- What sorts of feedback loops are typical in the production process?
- What methods of version control do studios employ?
- How do the various levels of personnel communicate during the production?

Each transcript was condensed into a shorter summary focusing on the participant's answer to each of the research questions. Finally, these summaries were broken down further and compiled into a list of questions, with a one-sentence paraphrase of the most important point stated by each participant. These paraphrased responses are included in section IV.2. Condensing and quantizing the data in this way allowed the researcher to quickly see the main patterns and differences in the responses.

III.2. Literature Search

Most studios are reluctant to divulge much information regarding their proprietary systems, but they sometimes share high-level information about processes, tools, or challenges related to their pipelines. Production studios often give talks or publish papers at the annual ACM SIGGRAPH conference. It is in these talks and papers from SIGGRAPH and other venues, such as the Eurographics conference and ACM Communications Journal, where most of the background literature related to this research was found.

Published papers that deal with production pipeline challenges, implementation, and concepts were mined for useful principles. Because the nature of production technology changes constantly, research focused on publications from the last decade, but important papers predating that time were found as well.

The Texas A&M University library system proved invaluable in locating papers and several textbooks for this research. The online catalog system and access to the full ACM digital archives provided a flood of material matching the search keywords. Much of this material was not relevant to the research - for example, a search for the keyword 'pipeline' returned 15,164 results, mostly papers on microprocessor architecture or embedded systems research. Refining the search by adding the terms 'animation,' 'production,' and 'project communication' yielded more useful results.

Many of the papers focused on a specific sub-pipeline, as mentioned by Alsup in section II.1.2, or discussed a software tool implemented to solve a production problem. While this information was useful as a *component part* of a pipeline, publications concerned with the larger aspects of communication, approval, feedback, and organization for digital productions were harder to find.

Fortunately, much of this perspective was found in the book *Producing Animation*

by Winder and Dowlatabadi [30], and SIGGRAPH presentations from Geiger [13] and others, which were discussed in section II.2.1.

III.3. Study of Current Systems

As mentioned in section I.2, the needs and goals of industry productions are not always the same as student projects. To apply any principles from industry practice to academics, an understanding of existing systems in academic programs must be obtained.

Having designed and built a project pipeline for students in the Texas A&M Visualization program, this researcher had full access to the details of that implementation and the consequences of its deployment. At the time of this writing, the Visualization Lab pipeline has been available to students for over one year. It has been used to create and manage a large number of projects, providing much data about successes and problems with the pipeline.

The Visualization department has an internal help ticket system which is used by students and faculty alike to report problems with systems and software. The help system has been in continuous use since 1996 and provides an enormous, detailed database to comb through. A search for the category tag 'pipeline' yielded over fifteen tickets ranging from bug fixes and feature requests to group projects and tool usage questions. These results and their implications for this research will be discussed in the next chapter.

III.4. Summary

The methodology and process for this thesis can be summarized in three parts:

- First, a series of interviews with seven different industry professionals was con-

ducted. Each of the participants was asked the same set of questions, and their answers to each were broken down into short summaries.

- Second, a search was conducted for supporting and related literature about animation pipelines and digital production management. This search was done through the Texas A&M University library system and on the internet. Relevant papers were categorized as relating to one of the following: pipeline definition, production management, tools and workflow, education, and open-source projects.

- Third, an example of a project pipeline for education was studied, combining personal experience from the development, deployment, and maintenance of the pipeline with a log of email-based help tickets. These contained suggestions and problems voiced by students about the pipeline tools.

CHAPTER IV

SUMMARY OF INTERVIEWS

The researcher followed a plan of interviewing participants from studios with diverse size, focus, and discipline. Seven interviews were conducted with professionals from feature animation, television animation, and video game studios. In addition, the researcher contacted and interviewed the creator of the previously mentioned openPipeline project to get the perspective of open-source projects in relation to industry practice.

Each of the interviews was conducted according to the project protocol approved by the Texas A&M IRB. Interviews lasted between thirty-five and sixty-five minutes. The following subsections present interviewee backgrounds and a summary of each interview. Full text transcripts are included in the Appendices.

IV.1. Background Info

The first section will discuss each participant's background, to provide some context for their experience relevant to this research.

IV.1.1. Interview 1 - Feature Animation Studio

The first interview was with a lead technical director at a feature animation studio. As outlined in the project protocol approved by the IRB, the interview started with questions about his background and how he came to work in his current position.

This subject studied symbolic systems in college, with a concentration in human-computer interaction. Symbolic systems is a multidisciplinary degree combining elements of computer science, philosophy, linguistics, and psychology. He initially

became interested in computer graphics through his work with a university virtual interaction lab. After graduating, he applied for a job as an apprentice Technical Director at his current studio, and eventually moved into the lighting department as a lead TD. For this job, he worked with the supervising TD to help decide the technology direction of his assigned film. His responsibilities allowed him to manage other TDs, but to also write code and develop tools as well.

This interview discussed pipeline concepts and the rest of the topics outlined in the project protocol, including technology challenges for films he's worked on and ideas for future pipeline tools. This interview served as a good starting point for thinking about pipeline concepts. This interview provided several helpful analogies for how pipelines should work and gave this researcher ideas to discuss with subsequent interviewees.

IV.1.2. Interview 2 - Multi-discipline Studio

This interview subject started out working technical support at a feature animation studio, doing everything from system administration type work to phone hotline support. Eventually he moved into software development and wrote 3D graphics translator tools to integrate CG imagery into 2D films.

He worked on some outside projects before moving on to a more supervisory role at another studio, where he oversaw pre-production sequence work for several films. He then went freelance for a while, doing consulting work for several clients. The up-and-down nature of the work led him to return to feature animation production as a CG supervisor, overseeing the design and construction of a new pipeline for his studio.

At his current job, he is now the manager of project engineering. His team manages central technology used by the studio's projects across multiple media platforms.

He was a big proponent of building pipelines and tools with the Unix philosophy in mind and much of the discussion centered around ways to uncouple parts of the pipeline and make things more modular.

IV.1.3. Interview 3 - TV Animation Studio

After earning his bachelor's degree in computer science, this subject worked at a large visual effects house for several years, building pipeline tools to help the studio handle larger productions. He then got a job at his current studio, which had never employed any programmers or people interested in putting together a real production pipeline.

The systems there had formed organically around the needs of the artists for many years, and were not well-defined. He came in and demonstrated the value of having programmers, and of setting up a pipeline structure to make artists more efficient. Very soon after that, more programmers were hired and he became the team lead. He eventually was given charge of all the technical means of the studio, encompassing the I.T., programming, and engineering departments.

His discussion of the protocol questions provided insight into the culture and approach of his studio, which typically has to turn around projects extremely quickly. Many pipeline decisions are based on what the artists need, and a conscious effort is made to keep instructions to the artists from feeling authoritarian.

IV.1.4. Interview 4 - Feature Animation Studio

This subject engaged in a broad range of academics as an undergraduate. While working on his graduate degrees, he volunteered at the annual ACM SIGGRAPH conference and made good contacts in the CG industry. His graduate research got him noticed by several studios, and he began working at a visual effects studio after graduation.

Working in visual effects proved to be exciting and challenging, but it took its toll on family life. After a long period of intense production crunch, he moved on to work at a feature animation studio. The saner working environment and keeping of every part of the production process in-house appealed to him. He has worked there ever since.

He further discussed some of the disadvantages of the visual effects business model, building pipelines for 80% of possible cases, and the evolution of pipelines over the years. This interview provided several useful concepts to the research, including some practical application of the Unix philosophy described in interview 2 in regards to the data model at his studio.

IV.1.5. Interview 5 - Open Source Project

The next interview focused on efforts building and growing the openPipeline project, which is discussed in greater detail in section II.4.

This subject studied biological anthropology in college, then decided to go to art school and get an MFA in a multidisciplinary program similar to the Texas A&M Visualization Program. He took a job at a feature animation studio soon after finishing graduate school and worked as a character TD on several of their films.

Eventually, he returned home to work on other projects. He wrote a textbook, started his own studio, freelanced at other studios, and did some teaching. It was during this time that he started forming the specification and ideas for openPipeline, eventually taking the project with him into academia when he was hired to run a new academic research lab.

After several years he returned to the same feature animation studio and spent a year and a half doing character setup (rigging) for one of their films. More recently, he moved into global development, creating tools and workflows to be used across all

shows and researching tools and technology for future shows.

Getting a more academic-centric view of the pipeline was crucial in forming the suggestions developed by this research. This interview provided good insight into the most important aspects of a pipeline for students.

IV.1.6. Interview 6 - Video Game Studio

This subject did his graduate work in computer graphics and then worked as a software engineer for several feature animation studios. He worked on several projects, including the CG rendering pipeline used for 2D films. His work moved closer and closer to production, making tools for the modeling and look development departments until eventually he became a lighting TD.

He then made the change to video games to write tools for the cinematics group, which makes all the cut scenes and in-game movies for the studio's games. Despite the fact that the content produced is very different, he found feature animation and video game cinematics to be very similar. Most of their current projects are shorter, so his role tends to focus more on artist support.

This interview provided more insight into the differences between large, long-term productions and smaller ones that finish just as the teams are hitting their stride and figuring things out. This subject also echoed some of the concepts mentioned in interview 2 regarding de-coupling the pipeline from data specifics.

IV.1.7. Interview 7 - Video Game Studio

This subject is the CG supervisor and head of pipeline and tools for a major video game franchise. He has the most artistic background of all of the interviewees. He graduated from college with an art degree, focusing on sculpture, but had his interest in CG sparked by one computer art class offered at the time.

He started doing cut scenes and animation for a small video game studio, followed by some work in the research division of a PC game studio. He created a lot of art assets for one of their early games and began to learn some of the basics of database programming to help organize all the artwork. He began doing more software tools, shaders, and generally technical work for their games before moving on to a different job as a senior technical artist at a large video game studio.

In the handheld games division, he wrote some tools to generate game assets, manage player profile pictures for sports games, and port geometry from one console format to another. Eventually he started working on the game engine for one of the studio's next-generation console titles, which needed to be upgraded from an older version to take advantage of the hardware technology on the new platform. When that project was complete, he shifted to his current position overseeing the tools for one of the studio's popular game franchises.

This interview provided a good look at how making games differs from films and animation, but also illustrated the crucial role of communication in all forms of digital production.

IV.2. Responses

As stated in Section III.1, each interview was broken down into a short summary and a further one-sentence paraphrase of the participant's answer to each question. The following subsections present these short summaries.

IV.2.1. Interview 1

This subject defined a pipeline as the 'secret sauce' of a studio - all of the project's processes and workflows, realized or embedded in software. He stated that a good

pipeline ensures the integrity of production data and facilitates collaboration between people. He cited email as the most-used communication tool at his studio because of its ability to be specifically directed to groups of people. Shot or sequence mailing lists allow relevant information to be sent only to those who need to see it.

He did not elaborate on version control methods, but did stress the need for studios to be able to retrieve specific configurations of shots. For example, a director may ask to see a shot the way it had been rendered five months prior. Under current systems, this request is very difficult to fulfill because all of the assets and their dependencies have changed. He described a version control system that might be able to achieve this through parsing more metadata *about* an asset.

On the concept of flexibility in a pipeline, he made an analogy to the layout of a grocery store. The aisles are placed based on market and social research, but as the store owners observe people moving around the store, adjustments are made to increase the visibility of some products or subconsciously guide shoppers to move through the aisles in a certain way. A pipeline should be flexible enough to change based on observations about the artists' behavior and their daily use of the tools.

At his studio, pipeline decisions for each film are made by a committee of TDs and supervisors. This group discusses the big-picture needs for the project and comes to a consensus on the major development initiatives that will need to be started. Smaller, more reactive decisions based on artist's needs are handled by individual TDs later on during production.

The biggest evolution of the pipeline at his studio was the expansion to incorporate stereo renders for 3D release of their films. Significant changes had to be made in the development, layout, and rendering workflows to create and finish shots in stereo without doubling the amount of work.

IV.2.2. Interview 2

Interviewee 2 had a broader definition of a pipeline than Interviewee 1. He saw it as a Venn diagram combining the circles of production management, creative (the artists), and technology. Production management runs the show, organizes the teams, and figures out how the work is going to get done. Creative makes the story, the art, the animation - anything that's going to show up on screen. Technology supports the artists' needs and the policies of production management. The intersection in the middle of all of that is the pipeline.

To him, a good pipeline reacts to the artist's conscious actions, automatically supporting them. He compared it to breathing - most of the time a person doesn't even think about it, but they can control their breath if necessary when the situation calls for it. The pipeline should be out of the way and let the artist focus on the creative work.

When asked about feedback loops and communication, he posited that email is inherently noisy and should take a back seat to other forms of communication. Databases, for example, allow a user to filter and find the information they want when they want it as opposed to getting an email every time something changes. Face-to-face interpersonal communication is better still, because it captures the full context of what's being said in a way that text-based notes cannot.

This subject took a more global view of data management not limited to version control, but expanded to a stack. The top level is policy, usually a decision between push or pull methods chosen by the technical leadership on the project. The next layer is the software tools artists use to do their work. Beneath that is asset management - for a particular circumstance, what versions of the assets should be combined to make a shot? The next layer down is version control. These systems should be

content-agnostic and handle only generic blobs of data, rather than expecting certain file types or storing anything other than file version history and logs for each file. The bottom layer is the hardware and software upon which everything runs, such as the operating system, file system, network location, and so on.

As with many of the research questions, this subject related his answer about pipeline flexibility to the Unix philosophy and good software engineering practice. The best way to be flexible is to leave the pipeline dumb and configuration-based so that it's not hard-wired to any specific production data. Meta-data, attached to assets or software tools, can be used to define things on the fly.

Relating to his earlier answer about data management and its top layer of policy, he described the hierarchy for pipeline decisions as a tiered system. The producer and executive producer define the expectations and needs of the project. The production manager, visual effects supervisor, CG supervisor, and technical supervisor get together to actually figure out how to execute those needs in policy and software.

Finally, discussing pipeline evolution and future systems, he predicted more movement toward databases as the backbone of a pipeline. However, systems using these databases should not be fully dependent on them. That way, if the database goes down for maintenance or technical problems, production doesn't have to grind to a halt.

IV.2.3. Interview 3

This interviewee defined a pipeline, broadly, as a communication technique between two different groups of people who have different goals and needs. The pipeline must mediate between those sets of needs and serve the requirements of individual artists. The best pipeline is the one that annoys the artist the least, because the goal is for the artists to be as creative as they can, as much as they can, rather than worry

about technical details.

Unlike other, larger facilities, his studio relies primarily on physical, face-to-face communication and eschews email almost completely. Unless they are very simple directions, text-based notes do not capture the full meaning and context of most feedback. This is especially true of feedback about animation. Having the director 'act out' his intent is much clearer than trying to accurately describe the same action in text.

Because of the speed of production at the studio, this interviewee and his team decided to enforce a 'moving forward' system of version control. There is no time to backtrack, so new versions of assets are automatically pushed out to artists. On shows with many assets, there is never more than one version of a given asset unless it is a main character.

The most pressing factor for his studio is limited production time. An important way for their pipeline to be flexible is to use only tools that can get quick and easy support. In almost all cases, this means writing software in-house because a third-party software company would not be able to meet the time demands required for the project. Being rigidly confined to programs that someone else wrote, and being unable to change or fix them, is an unacceptable scenario when the project has to be completed in days or hours.

He approached pipeline decisions from a more artist-driven position. Because of the studio's small size, the speed of production, and the culture of the workplace, any decisions handed down about pipeline or workflow need to be given very carefully so as not to feel authoritarian. Some details, like naming conventions, are decided by tech leadership at the top level, but nearly everything else comes from conversations with the artists about what annoys them and what would help them work more efficiently. His studio also has the benefit of very low employee turnover, so their systems can be

more deeply tailored to individual needs. They don't have to train a new employee every few days or weeks to learn the pipeline.

Building the pipeline in a small-problem, piece-by-piece way was crucial to his opinion of pipeline evolution. He felt throwing out the existing system usually causes more headaches than it's worth. Instead, the pipeline designer(s) should focus on fixing small, solvable problems with an eye towards the eventual completed system. Each component should also be able to pass data in and out of the rest of the pipeline so that they can be swapped in a modular way.

IV.2.4. Interview 4

This subject defined a pipeline as the backbone of the production process. It's a way of establishing a structured, compartmentalized approach to tasks to make things more efficient and carry a shot from a bunch of drawings on paper out to the final rendered frames. Every pipeline has its strengths and weaknesses, but the best pipeline is one that handles 80% of the work effortlessly, which gives you the time and resources to deal with the other 20% of cases.

On the subject of feedback and communication, he agreed that email is noisy and intrusive, but added that the intrusiveness is sometimes a good thing. For example, as a technical supervisor, he receives upwards of six hundred emails a day, so writing mail filters becomes critical for filing away messages that don't require immediate action. Anything not captured and archived by the filters demands attention, and sometimes it's necessary. He also echoed Interview 2's point about allowing users to filter and find the information they want - through web forms, graphs, and even iPhone applications that let them check the status of renders.

He described how asset management and version control are taken completely out of the artist's hands at his studio, which has moved toward a true 'push' system

where the latest asset versions are automatically pushed out to artists. If the new version breaks something, then it breaks for everyone, and the asset can be reverted or fixed quickly. Old versions are maintained for archival purposes.

The key to a flexible pipeline, according to this interview, is the ability to integrate new techniques and data into the process, sometimes in the middle of a production. For example, on a recent feature animation project, the layout department decided to start incorporating captured hand-held camera work into their shots, even though the project had not been set up to use that technology. This change required the engineers to write new software to handle the capture data and put it in a format that worked with the existing pipeline. At this studio in particular, last-minute story changes may introduce a new special effect or visual element that the current pipeline does not have a way of creating. In those instances, new tools have to be written to achieve the desired effect in service of the story.

The people responsible for making the decisions about *how* to make these changes and integrate new technologies are the visual effects (VFX) supervisor and the technical supervisor. The VFX supervisor's job is to interpret what the director and art director want to figure out how to make that happen from a production standpoint. The technical supervisor works with the VFX supervisor and advises them on the process, pipeline, and tools that will help achieve the director's vision. These two supervisors then define the rest of the leadership structure based on how best to approach the project. Some pipeline decisions (such as naming conventions) are decided at this level, but there is always input from the production departments who may have suggestions about better ways to work.

The biggest evolution in the pipeline during his time at the studio was the decoupling of the data model from the shot workflow. Initially, the shot data and assets were passed along in an assembly-line fashion, which led to some complications in

the production because the data had changed hands so many times. Eventually they determined that a hub-and-spoke model would be more stable. In this model, all of the show's assets are stored in a central repository and are checked in and out as necessary. This made it easier for departments to work out of order without causing problems for each other.

IV.2.5. Interview 5

This subject described the pipeline as the glue between all the artist's stations in a studio. It's the software and processes that brings in the elements an artist needs to get started on their work. The pipeline then takes that work data, converts it, and sends it off in a usable format for other people in the facility. Most of this is happening behind-the-scenes. The artists don't see this when they work. A good pipeline has a well-defined structure, but is modular in construction so that pieces can be swapped out. It's scalable so that it can handle varying volumes of data and users.

On feedback loops and communication, he said that email lists were an effective way to reach the appropriate parties with questions or problems, but the best feedback usually comes from direct, verbal communication with the artists. Co-location with the artists is important for software engineers because they can get better information than a simple bug report, or even look over an artist's shoulder as they work to see what they're doing to cause an error.

According to him, a studio's version control strategy depends on their priorities and available resources. A visual effects studio serving an external client may need to be able to give the client whatever shot version they ask for, from any point in time. A feature animation studio that is its own client may not care about older versions because they're focused on moving forward with whatever changes the director asks

for. They may store older versions for archival purposes, or to restore if something goes wrong, but they don't have any intention of going backwards in versions.

According to him a flexible pipeline is one that is less aware of the specifics of what data actually means. The pipeline simply carries data from one location to another, and can respond to queries about its contents and the software required to handle it. A pipeline that looks for a defined number of things, of a specific data type, in a specific hard-coded location, will not be able to handle any change or expansion very well.

Those kinds of pipeline decisions are usually a power struggle between the studio at large and a specific production. Decisions about tools and process will be either studio-centric or show-centric. If technical leadership is on the studio side, then each production uses the tools and policies dictated by the studio. If an individual production has some specific demands for resources or tools, development can be driven by the needs of that production. Ideally the facility can find a position somewhere in the middle, where the studio has a coherent global strategy for the pipeline, but individual shows can ask for and try new things to improve on current systems.

This back-and-forth between the studio and the shows drives the evolution of the pipeline. Reacting to problems on previous productions creates new tools and methods, which may or may not carry over onto the next show depending on its needs and the success of the previous project.

IV.2.6. Interview 6

This subject's definition was similar to the answer given in Interview 2. A pipeline is a combination of the production's *ecosystem* and *culture*. The ecosystem is simply the structure and framework in which the production lives - the show's hardware, software, personnel, and processes. The culture of the production is the rules of

behavior - the policies that dictate how the project will be done. A good pipeline handles version control, asset management, and communication in a clear and simple way that does not confuse the artists.

Artist notes are incredibly important to communication in a production. They help chart an asset's history and enable asynchronous interaction across departments or even across time zones. Free-form notes unrelated to revisions, such as to-do items or warnings about potential problems are important as well. These kinds of notes should ideally trigger an email to get people's attention when things change. An even better system would be an integrated operating framework that allowed artists to see changes to data in real time, similar to a day trader watching stocks.

Version control for openPipeline was made deliberately simple - artists simply save their work and a new version is automatically created in a defined location. openPipeline keeps the last ten versions and has functionality to recall old versions if the user wishes. For small-scale projects, a web-based revision control system can be used to store assets and work collaboratively with off-site artists.

Pipelines should be flexible - to a point. They need to have a set of basic, rigid rules and be flexible in small, smart ways. For example, generating metadata about a shot or asset that's easy to read by multiple software programs. Once production begins, there can't be a lot of flexibility in the pipeline. The aim is consistency, and any customizations or adjustments for flexibility need to be made with care.

In production studios, decisions about pipelines are made after many meetings between TDs and production supervisors. They are confirmed or adjusted by department heads who are in tune to their department's needs. The top-level supervisors have a more global view that might not necessarily be available to a specific department concerned with its own corner of production.

Over its lifespan, the openPipeline project has evolved from a specification to a

Maya implementation in MEL to a host of other implementations for different software programs with support for a myriad of data formats. It has grown from use exclusively at small studios and academic institutions to mid-sized production studios all across the world. It has even formed the basis for some notable commercially-produced asset management systems.

IV.2.7. Interview 7

This interviewee differentiated between pipeline and workflows in his answer. According to him, pipeline is the tools or applications for creating the game's assets and putting them in the right format for the console or destination platform. Workflow involves the artists and approvals systems, and the processes that guide assets from concept to final product. To him, the word 'pipeline' means technology and 'workflow' means the human factors and policies used to get the project done. A combination of these two things is required to get a game made.

A good pipeline is dependent on a studio culture where it's okay to speak up and identify problems. Having people who voice their opinions about solutiions and don't accept temporary workarounds is key to building strong, robust tools. A good pipeline does not break easily, and is well-documented.

This subject felt that the communication and feedback methods used in production depends heavily on the team and how well they are functioning. Video games tend to have less stringent feedback loops, but rather broad art direction for the entire build of a game. Face-to-face communication is great because it's immediate and delivers the full context of that feedback. But, it's not documented the way email is. Being able to go back and refer to previous communications through email is useful. He described an ideal 'living document,' an editable wiki-style page or resource that documents solutions to problems and points out people to ask about certain issues.

Having assets logged with good version notes is important, and the same goes for software development and deployment. If a pipeline tool is updated and new problems appear, the engineers can look to a particular version of the code to see if a bug was overlooked before deployment. His studio makes use of relational databases to store assets and their connections, so that if one asset changes it automatically affects any child assets in the database.

Pipelines for video games are usually very production-specific, unless they make use of broad, well-documented engines such as Unity or Unreal. A studio can make one game with one pipeline because the tools are built for specific production challenges, and flexibility outside of that is a low priority.

Decisions about the pipeline are a collaboration between the run-time engineers making the tools and the artists creating the game. The engineers need certain things to make the game run, and the artists generate assets and data that have to be conditioned in a certain way to run in the game. The challenge comes in figuring out the easiest and most direct way to make that translation. Any tools that are developed have costs - they must be maintained and documented. As the CG supervisor, he has the final say on which tools are worth the time and effort.

Video game pipelines have evolved exponentially in their complexity and performance to keep up with the demands of newer games. At his old jobs, games did not have many assets, so keeping track of everything was not an issue. Now a game may have thirty or forty different asset types and many in-house software tools and packages for the graphics cards needed to handle increasingly complex geometry.

CHAPTER V

RESULTS AND DISCUSSION

This chapter discusses the general principles discovered during the course of the research. Educational environments could benefit most from applying these principles.

V.1. Recurring Themes from Interviews

When asked to define a pipeline, each interviewee gave different answers, sometimes using creative analogies to describe what they thought of as a pipeline. However, nearly all of them considered a pipeline to be a combination of, or the glue between, all the areas involved in the project. Production management, creative (the artists), and technology must work together to communicate, establish structure, and meet the artistic and budgetary needs of the project. Interview 2 even described the pipeline as the intersection of a Venn diagram of these three main areas.

When it came to describing a 'good' pipeline, two main threads of conversation emerged. First, a good pipeline should automatically support what the artist is doing without requiring any of their time or energy. Artists shouldn't have to worry about where to save their files, or how often, or what to do when they're finished working. The behind-the-scenes tools and the chains of communication/approval should reflect this. Second, a good pipeline should be designed and structured in a modular way so that it's simple to swap pieces out without drastically affecting the rest of the pipeline.

Interviewees were highly divided on the subject of email as a method of communication and feedback in a pipeline. Some, such as Interview 3, stressed the richness and context of physical communication, especially in the area of feedback for artists.

A director physically acting out what he wants in a character performance is more useful for an animator than text-based notes. On the other hand, email is ubiquitous, and can be automatically triggered by the system to notify artists when something needs their attention. Group mailing lists (for artists working on the same sequence or shot) can be used to direct feedback more locally without filling up everyone else's inbox with irrelevant messages.

Version control strategy is defined at the management level and depends on the studio's priorities and available resources. For example, Interview 1 brought up the scenario of a director wanting to recreate a shot from six months back. If the ability to do that is important to the studio, then systems will be put in place to store and track all those old versions of assets so that they can be called up at a moment's notice. Like other pipeline tools, versioning should be taken out of the artist's hands and performed automatically.

Two main points emerged from the discussion of flexibility in the pipeline. First, whenever possible, use software tools that can get quick, responsive tech support. Relying on external third-party support can be problematic under a tight deadline. It is for this reason that most studios write their own proprietary tools. Second, use metadata like scene descriptions or configuration files to keep the pipeline more general, rather than closely tying the tools to data. Interview 2 used an example where his studio had some very complicated shots with over ten thousand assets. Rather than trying to reference each asset into the shot file, they used a text file to define which objects belonged in the scene and then only loaded the ones they needed at any given time. Keeping things generalized with scene descriptions allows the studio to readily handle shots of varying size.

The decision-making process for pipelines varies from studio to studio, but for the most part the answers from each participant were consistent. Decisions about

versioning, push vs. pull, and leadership and approvals structure are typically made by upper-level management including the producer, visual effects supervisor and technical supervisor. This group has a more global 'big picture' view of the project compared to a department supervisor, who may be most concerned with his department's immediate needs. However, this does not mean that everything is handed down in a forced, authoritarian manner. The needs and suggestions of each department are taken into account to give everyone an agreeable set of processes to follow.

The final interview question covered the evolution of pipelines. Discussion on this topic was all over the map. Interview 1 talked about the increased importance of stereo for CG productions, and the necessity to plan and optimize for stereo images. Interview 3 charted the progress of his studio from having no established pipeline, to implementing components one at a time, to having a very efficient set of tools and processes that allow very fast turnaround on projects. Interview 4 spoke about the process of decoupling the data model from the shot workflow to eliminate dependencies and bottlenecks. All of the answers, though, could be boiled down to a search for greater efficiency in production.

V.2. Findings from Publications and Literature

The literature search portion of the research yielded nearly forty references across many publications. Articles and papers from SIGGRAPH, SIGCHI, SIGCSE, ACM, IEEE, Computer Graphics World, and published books on animation production provided useful insight into the various topics of this thesis.

One important insight provided by the literature search was the critical role of the producer and top-level studio leadership in forming a pipeline. Winder and Dowlatabadi's book gave the most detailed look at the organization and direction

required to assign leadership and parcel out labor in a way that will get the production finished on budget. While academic pipelines do not necessarily contain a 'producer,' establishing responsibility and chain of command is important to ensure that systems are maintained and updated to meet student needs.

Scalability is another important concept touched on in both Chuang's and Geiger's SIGGRAPH presentations [7] [13]. Setting up a pipeline that can be easily scaled is necessary in academic environments, where students may work on individual projects or collaborate with others in teams. Geiger in particular emphasized the concept of *people* being the most valuable part of any pipeline. Allowing students to easily collaborate taps into this value.

The capacity for clear, constructive feedback is a crucial part of the pipeline. This is especially true when artists have to deal with others who have differeng expectations or disparate levels technical knowledge. Hirsch's SIGGRAPH presentation [15] about delivering creative feedback, as well as Phalip's paper on communication challenges in film scoring [22], were both useful in this regard. Phalip offers an interaction design approach to bridge the gap between collaborators with different backgrounds, and to address ambiguities in communication. This approach was helpful in formulating suggestions for communication tools in an academic pipeline, as was Fussell and Weisband's research into instant messaging as a tool for multitasking and collaboration [12].

The majority of publications found dealt with creating tools to meet specific production challenges. These have already been covered in Section II.2.2, but the overarching point of the papers remains - that a pipeline should be able to accommodate the integration of new tools and data in order to meet production needs.

V.3. Existing Academic Pipelines

Re-examining the Visualization Lab pipeline in the context of this research provided fresh perspective on which parts of it worked, and which needed improvement. The biggest problem for students seemed to lie in the versioning system. Students are accustomed to using the 'File–Save As' dialog in Maya to create a new incremental version of their work when they reach a significant milestone. The workflow defined in the Lab pipeline required that they save their work and then use a command line tool to manually perform a backup with some check-in notes. This extra step, along with the introduction of the command line interface, was initially confusing to students and led to the creation of duplicate project folders in their work areas.

The Visualization Lab pipeline requires students to type in commands for tasks like asset and shot creation, and for starting their Maya sessions. However, the utility of making artists use the command line to perform tasks is open to debate. If the goal of an education program is to prepare students for industry, and industry does not require artists to use the command line, do students reap any benefit from having to learn to use it? Both Interview 3 and Interview 4 stated that artists do not use command line tools at their studios, but instead use GUIs for all their tasks. Command-line versions of these tools exist, but mostly for debugging and testing purposes. They are not used by artists.

Most projects in the Visualization Lab are individual assignments where only one student is modifying assets and tracking his or her progress. However, for cases like the summer industry course or some directed-study projects, group access for small teams may be required. Team members are added to a Linux user group and given access to a project directory. The structure and tools for groups are the same as for individual projects, and the only group-specific features are utilities to fix file

permissions in the project creation and shot publishing commands.

Studying the log of help emails was also useful. Some of the pipeline features regularly used by students, including the 'make asset' command, were not part of the initial pipeline distribution but were suggested by students and incorporated later.

V.4. General Principles

Based on the results from interviews, study of published works, and experience developing a small project pipeline for academics, pipeline practice can be broken into the following five principles: stability, clarity, unobtrusiveness, modularity, and generality.

V.4.1. Stability

Stability in a pipeline is most closely related to the technology used for production. It means maintaining data integrity, providing working tools, and functioning in consistent way that does not surprise its users.

There are two key steps to ensuring stability in an academic pipeline: sanity-checking input, and software testing. Sanity-checking input is a basic software development practice and is easy to implement. A sanity check is a basic test to see if the input is rational and rule out an obviously incorrect results. For example, if a program accepts only numeric integers as input and the user enters a string of letters, the input is incorrect and the program should output an error message. This type of check is typically placed at the beginning of a software program before other operations, so that no calculations are wasted on incorrect input.

Testing, however, can be more difficult. Most academic facilities do not have the resources to do extensive testing so most effort is usually spent on fixing bugs that

are reported by students.

In this respect, academic environments are most like the situation Interview 3 described, where his team takes summers to write and test new tools. Individuals responsible for academic pipeline development should take advantage of breaks between semesters to test their tools and identify as many bugs as possible. This testing time is important, because if something breaks during student production that interferes with their work, it puts a burden on both the students and the support staff.

Money is not typically at stake when delays are caused for students, but time is. They do have project deadlines to meet. Time wasted fixing errors prevents students from engaging in more creative work, and prevents support staff from developing new tools or implementing new features.

V.4.2. Clarity

Clarity involves both the software and personnel aspects of a pipeline. Data flow should be well defined so artists know what they should expect to receive as input and what to deliver as output. Approvals should be well structured so artists know who to report to for feedback about their work. The feedback itself should be well delivered so that ambiguities are minimized. Tool interfaces should be well designed so they can be used easily by artists. These tools should also be well documented so artists can find information quickly if they have a problem.

Most projects in academic environments are done by individuals. Students handle all aspects of production themselves and do not have to worry about passing data to others (though there are exceptions in the case of group projects). It is important to instill good practice by defining the flow of data through any pipeline students use. Having a well defined set of steps that makes sense is more important to students than the details of push systems versus pull systems.

The approvals structure in academic pipelines is straightforward and takes one of three forms: instructor approval, self approval, or team leader approval. The most common form of feedback loop is the one between the instructor and the student, and sometimes between students during class critiques. Making instructors and students aware of feedback techniques such as those presented by Hirsch can lead to better critique sessions and more useful feedback.

Interface design for academic pipelines is a struggle between GUIs and command-line tools. The preference for one or the other comes down to the goals of the institution. Does that program see a benefit to making students use the command line, or do they prefer familiarity and ease of use? Based on the interview responses, and to ensure usability for all students, GUIs should be created where possible. Using GUIs allows the pipeline designers to limit user access to data and decreases the chances of student errors through incorrect input or careless clicking. As mentioned by Interview 4, ideally the commands performed by the GUIs can also be run from the command line for testing purposes, or by users who prefer to do so.

All of these aspects must be well-documented so that students (and instructors) can find information about them quickly. The easiest way to implement this is through an editable wiki-style web page. Command lists, articles, how-tos, screen shots, and code snippets can all be accessed and updated on a wiki through simple interfaces.

V.4.3. Unobtrusiveness

An obtrusive pipeline that gets in the students' way or makes them do extra work is antithetical to the *purpose* of a pipeline. An academic pipeline should allow students to spend the maximum amount of time creating and working instead of dealing with technical details or repetitive, insignificant tasks.

This *un*-obtrusiveness can be achieved by taking certain tasks behind the scenes

for students. For example, one of the strengths of openPipeline is its automatic handling of file versions when users click the 'save' button in Maya. The file is saved with a new version number in the correct project location, without making the user perform any extra steps. The 'make shot' command in the Visualization Lab pipeline is another example of this behind-the-scenes work. Users simply select the assets they want added to a shot, and all the file creation and referencing is performed for them automatically.

Being unobtrusive carries over to production tracking, as well. Students should not be bombarded with emails at every change or publish. A better approach, and one that fosters more collaboration, might be to generate data that could be easily imported into a Google Doc spreadsheet. That way students can open a tracking sheet and see any changes that have been made.

V.4.4. Modularity

The principle of modularity is a tenet of the Unix philosophy and focuses on making software cleaner and simpler. Eric Raymond, in his book "The Art of Unix Programming," [24] wrote:

> The only way to write complex software that won't fall on its face is to build it out of simple modules connected by well-defined interfaces, so that most problems are local and you can have some hope of fixing or optimizing a part without breaking the whole.

As a pipeline increases in complexity, the need for 'well-defined interfaces' between the tools becomes greater. Rather than writing a large program to perform multiple tasks, pipeline designers should break these tasks down into individual programs and ensure that they can pass data easily between them. When implementing

a modular approach, the designer should ask: if one of the pipeline tools was updated or replaced, would all the other tools still work?

Academic pipelines do not have the benefit of a team of engineers working year-round to improve and update their tools. Since students come and go, having a set of organized, easily-connectable tools makes it easier to continue development even after the original designer has graduated or moved to another institution.

V.4.5. Generality

This principle could also be called 'reusability.' Building a pipeline that is re-usable across projects can help students be productive more quickly since they do not have to learn new tools. In the case of academic pipelines, generality is crucial because it is unlikely that the school or the students have the time or resources to build pipeline tools and protocols for every individual project.

An academic pipeline should be suitable for many *kinds* of projects, not just animation. Multi-disciplinary academic programs often encourage many different types of projects ranging from live-action films to animation to data visualization to interactive pieces. Handling many different types of project data requires a pipeline that is *content agnostic* and not tied to the specifics of data, as described in Interview 2.

Part of being generalized is being scalable. Academic pipelines should be able to scale from individual students to small or mid-sized teams. Group projects require special considerations for file permissions. Specific implementations depend on the operating system running on that institution's computers.

CHAPTER VI

CONCLUSION AND FUTURE WORK

Developers and interested students should be able to use the results of this research to build their own pipeline, or contribute to existing projects like openPipeline. In places where a production pipeline already exists, new tools or interfaces could be added based on the presented suggestions. The work presented here could also inspire other master's theses concerned with implementing and testing these suggestions in a substantive way.

Digital production is increasingly globalized, with studios sometimes having several international branches or subsidiaries. Working with an overseas facility presents new challenges for pipelines, such as how to sync and manage production assets between studios. This thesis does not discuss how studios handle remote collaboration. An investigation into these practices would be a useful extension of this research.

REFERENCES

[1] I. A. Alsup, "#0020: The technology is not the pipeline," Aug. 2009. http://cgsupervisor.blogspot.com/2009/08/0020-technology-is-not-pipeline.html

[2] N. T. Bajandas, "A post-mortem analysis of production process: The bricklayer's disaster," Master's thesis, Texas A&M University, College Station, TX, USA, December 2011.

[3] D. E. Bettis, "Digital production pipelines: Examining structures and methods in the computer effects industry," Master's thesis, Texas A&M University, College Station, TX, USA, May 2005.

[4] R. Bredow, D. Schaub, D. Kramer, M. Hausman, D. Dimian, and R. S. Duguid, "Surf's up: The making of an animated documentary," in *ACM SIGGRAPH 2007 Courses*. New York, NY, USA: ACM, 2007, pp. 1–123.

[5] G. Butler, A. Langlands, and H. Ricklefs, "A pipeline for 800+ shots," in *ACM SIGGRAPH 2008 Talks*. New York, NY, USA: ACM, 2008, p. 72.

[6] E. Catmull, "The problems of computer-assisted animation," in *Proceedings of the 5th annual conference on computer graphics and interactive techniques*. New York, NY, USA: ACM, 1978, pp. 348–353.

[7] R. Chuang and D. g. DeBry, "Creative collaboration: Effective CG pipelines: any size, any place," in *ACM SIGGRAPH ASIA 2009 Courses*. New York, NY, USA: ACM, 2009, pp. 1–68.

[8] P. Comninos, L. McLoughlin, and E. F. Anderson, "Educating technophile artists: Experiences from a highly successful computer animation undergrad-

uate programme," in *ACM SIGGRAPH ASIA 2009 Educators Program*. New York, NY, USA: ACM, 2009, pp. 1–8.

[9] D. S. Ebert and D. Bailey, "A collaborative and interdisciplinary computer animation course," *SIGGRAPH Comput. Graph.*, vol. 34, no. 3, pp. 22–26, Aug. 2000.

[10] L. Estebecorena, N. Sepulveda, and K. Sprout, "Rango: A case of lighting and compositing a CG animated feature in an fx-oriented facility," in *ACM SIGGRAPH 2011 Talks*. New York, NY, USA: ACM, 2011, p. 16.

[11] D. Fagnou, "Stereofx: Survey of the main stereo film-making techniques," in *ACM SIGGRAPH 2011 Studio Talks*. New York, NY, USA: ACM, 2011, p. 6.

[12] S. R. Fussell, S. Kiesler, L. D. Setlock, and P. Scupelli, "Effects of instant messaging on the management of multiple project trajectories," in *Proceedings of the SIGCHI conference on human factors in computing systems*. New York, NY, USA: ACM, 2004, pp. 191–198.

[13] K. Geiger, "Keeping your money on the screen & off the floor," in *ACM SIGGRAPH ASIA 2009 Courses*. New York, NY, USA: ACM, 2009, pp. 1–51.

[14] D. Heckenberg, D. Gray, B. Smith, J. Wills, and C. Bone, "Quill: Birds of a feather tool," in *ACM SIGGRAPH 2011 Talks*. New York, NY, USA: ACM, 2011, p. 34.

[15] E. Hirsch, "Delivering creative feedback," in *ACM SIGGRAPH 2012 Courses*. New York, NY, USA: ACM, 2012, pp. 1–10.

[16] D. Maskit and C. Fong, "Production-grade scene translation pipelines," in *ACM SIGGRAPH 2003 Sketches & Applications*. New York, NY, USA: ACM, 2003,

p. 1.

[17] A. McNamara, J. G. Montalvo, D. Walvoord, and M. Friedman, "Revolution - evolution: The collaboration forges on," in *ACM SIGGRAPH 2011 Studio Talks*. New York, NY, USA: ACM, 2011, p. 5.

[18] G. Meeres-Young, H. Ricklefs, and R. Tovell, "Managing thousands of assets for the *Prince of Persia* city of Alamut," in *ACM SIGGRAPH 2010 Talks*. New York, NY, USA: ACM, 2010, p. 30.

[19] N. D. Naugle, "BIM principles to practice: Using BIM to create a new model for producing animation," Master's thesis, Texas A&M University, College Station, TX, USA, December 2011.

[20] R. O'Neill, "Building the perfect production pipeline one student at a time," *Computer Graphics World*, vol. 32, no. 8, Aug. 2009.

[21] R. O'Neill, P. Mavroidis, and M.-H. Ho, "openPipeline: Teaching and implementing animation production pipelines in an academic setting," in *ACM SIGGRAPH 2007 Educators Program*. New York, NY, USA: ACM, 2007.

[22] J. Phalip, M. Morphett, and E. Edmonds, "Alleviating communication challenges in film scoring: an interaction design approach," in *Proceedings of the 19th Australasian conference on computer-human interaction: Entertaining User Interfaces*. New York, NY, USA: ACM, 2007, pp. 9–16.

[23] Pixar, "How we do it," April 2011. http://web.archive.org/web/20110414200925/http://www.pixar.com/howwedoit/index.html#

[24] E. S. Raymond, *The Art of UNIX Programming*, 1st ed. Boston, MA, USA: Addison Wesley Professional Computing Series, 2003.

[25] A. Rempel, D. Broadland, S. Struben, and F. D. Fracchia, "Effective asset management for episodic television and features," in *ACM SIGGRAPH 2003 Sketches & Applications*. New York, NY, USA: ACM, 2003, p. 1.

[26] B. Robertson, "Disney lets CAPS out of the bag," *Computer Graphics World*, vol. 17, no. 7, pp. 58–64, Jul. 1994.

[27] "Disneyland, episode 63: Tricks of our trade," Walt Disney Studios, Feb. 1957, premiered on ABC. http://www.imdb.com/title/tt0833037/

[28] R. Weinberg, "Producing content producers [computer animation]," *Communications Magazine, ACM*, vol. 33, no. 8, pp. 70–73, Aug 1995.

[29] J. Whittington and K. J. Nankivell, "Group projects: Issues and practices in computer graphics technology," in *ACM SIGGRAPH 2004 Educators Program*. New York, NY, USA: ACM, 2004, pp. 3–4.

[30] C. Winder and Z. Dowlatabadi, *Producing Animation*, 2nd ed., T. Miller-Zarneke, Ed. New York, NY, USA: Focal Press, 2011.

[31] U. Wolz and S. M. Pulimood, "An integrated approach to project management through classic CS III and video game development," in *Proceedings of the 38th SIGCSE technical symposium on computer science education*. New York, NY, USA: ACM, 2007, pp. 322–326.

APPENDIX 1

INTERVIEW TRANSCRIPTS

INTERVIEW 1

BRANDON: I'm here with Interview 1 from Studio Alpha and, um, we are gonna talk a little bit about pipeline stuff. Uh, the point of the questions and the point of the thesis is really to investigate different pipelines at different studios and see what the similarities are, what the differences are...how they compare and especially across different mediums like video games or television, movie production and visual effects. So the idea is to look at all of them and see, um, to assemble, I guess the best parts from each and figure out what's common so that I can take those pieces and hand it to somebody who may be building a pipeline for an open-source project or for an academic institution or whatever so, um, that's kind of the idea. So, um, since you're the first person and you're working in feature animation, talk to me a little bit about your background. What is your position at Studio Alpha and how long have you been there?

INTERVIEW 1: My background, um, so I graduated from [university] with a bachelor of science from, uh, in symbolic systems, which is an interdisciplinary major combining computer science, philosophy, linguistics, and psychology, so it's like it's kind of like a cognitive science. Um, you can choose multiple concentrations like natural language processing, artificial intelligence, human-computer interaction - I did HCI, um, and I did a little bit of work, how I got into CG is I did a lot of psychological lab work in this virtual human interaction lab, it was all virtual reality research. Um, and it was a lot of fun, and then I applied for a job at Studio Alpha as an apprentice TD. Um, and all TDs at Studio Alpha are basically pipeline people. They can be in different parts of the pipeline like different departments, but they are not image-producing people in movies, so we're unique that way compared to other studios. And so I started as an apprentice TD and then became a lighting TD, um,

was a lighting TD for about four or five years, and then, and I've been there now for almost, I guess I would say yeah almost seven years, and so and I've been a lead TD for um, I got promoted to a lead TD and then I've been there for about three years now, and I just wrapped up Big Film Two which is my first major production as a lead TD. And so, um, as a lead TD, we basically, we work directly under the supervising TD which is, he's the, or he or she is the head of the department for the production and they make the, the major decisions in terms of um, the technology direction of the show and all of the major initiatives of development that needs to happen. The lead TD gets to co-manage with the supervising TD but gets to be the lead, the tech lead of those projects. So, um, I really enjoy the position because it lets me manage, but also program and develop, um, not spend all my time in meetings. And so, um, and then uh that's basically the position.

B: Cool. Well that's, yeah that is nice that you get to do a little bit of both, you know you're not just, not just stuck behind, behind a desk the whole time or something like that, so. Um, great! That's, that's good, we know a little bit about um, your background, so I guess we'll move into the, more of the meat of the questions that we're looking at. Um, so you've talked a little bit about your position there. What, in your experience, at Studio Alpha in the almost seven years you've been there, how would you define what a pipeline is?

I1: Um, and, what's great is that you've already heard some of these answers, but um, a pipeline is, in like one sentence, is like the secret sauce for a studio. Um, it's basically all of the different processes and workflows um from beginning to end all realized or embedded in software. Um, that that's basically a one-sentence definition of what I believe a pipeline is.

B: [writing]...embedded in software.

I1: Yeah.

B: So there's, um, a tool-based element and also a time-based element, is that right?

I1: Time, workflow, procedure, I mean so many dif – like, the pipeline is basically how one thing gets from one place to another. All embedded and realized in tools or software or workflows.

B: So, now that we have sort of a rough definition, what, what, so you have a pipeline that starts things, you know, you have your assets and things like that and they move towards the final production of the shot. What makes a good pipeline?

I1: There's so many things that make a good pipeline. Um, depending on how big your project is, because that will better inform how, the different factors of what makes a good pipeline. Um, you basically want to make sure that all of your assets and your data, that, that your pipeline ensures their integrity, um, they can't be you know overwritten, um, no matter how many artists are working on the same data at the same time. Um, you have to be able to facilitate, uh, smooth collaboration between all sorts of different people working at the same time in different parts of the pipeline, um, if you're obviously on the same project. Um, the pipeline's all about you know organizing huge amounts of data, because you're gonna have, depending on how large your project is, lots of data, and you're gonna need to have each person be able to quickly access different parts of that data easily. Um, and that's definitely a huge challenge the bigger your project gets. Um, another thing is that a good pipeline has to be, has a has to have a solid enough foundation but yet be flexible enough to be adaptable to new technical needs or new creative needs, um, because you, you're not gonna want, if you're in a big studio you're not gonna want to reinvite a, reinvent a pipeline every time you start a new project. You want to build a foundation and then, if you move from one movie to another, you have that foundation, then you can, you know, enhance it, you can develop on top of it, change it a little, just for

the specific creative needs for the new project.

B: Right.

I1: Um, I think the other things are, you know, I I've kind of alluded to this, but it just can't get in the way of artists. It has to, the whole part, the whole reason a pipeline exists is to make things more efficient, so, um, you know, in, in my career I've discovered that that that goal is never gonna be 100% complete because, I mean, everything is always being improved upon. I mean in every industry and in every part of life, life, right? So, and if it was 100% solved, I wouldn't have a job, so...

B: Right. [laughs]

I1: Um, but, but yeah, I mean that, those are basically the, the top top things I would say a good pipeline needs to have.

B: Cool. Yeah and you talked about, um, having to have multiple people working on the same data, and you know not tripping over each other, not overwriting things...um, so what kind of feedback loops or communication tools make that possible, so that artists aren't always redoing the same things or overwriting each other's stuff.

I1: Yeah. It sounds prehistoric, but email.

B: [laughs]

I1: Email's really good. Um, asset tracking software of course. Um, you know it's interesting because there are definitely, different studios, I mean in my limited experience because I've only worked at one studio, but from what I hear from colleagues who've worked at other studios, um, just so many different people have different types of asset tracking software that scale differently, and ours doesn't scale as well as we'd like, and we're definitely developing new software to, you know, adapt to larger scale projects. So we really do rely on a lot of email. Um, we rely on things like, um, they're sometimes called command flows or action flows, um they're, there are certain types of

basic tools that are unit-based that are, um, specialize in deliveries and handoffs from department to department, so um, we're constantly reevaluating whether or not our system of, you know, handing off deliverables is a push system or a pull system. And I'm sure, like, every single studio goes through this challenge, um on, from project to project because some departments prefer you know, pushing something out you know like, "here's my deliverable, you've got new models for something, you take them." um, or some people just want a pull system where they, they put it in some staging area and then the person who needs it just pulls it whenever they need it.

B: Ok.

I1: Um, and so that's a constant challenge. You might, hopefully you'll hear about that a lot, with, as you go through your interviews, but, um, those are types of feedback loops that, that really help and they're constantly being developed. I mean, I can't even say if they're even good enough yet, like they're, they're just always being developed, because there's always these human factors that you can't predict, that these tools can't predict. So that's why I always go back to email, because email's like the easiest thing to do because you're like, oh, this is totally, this is ready, this is final, you can take it. Or, this is not yet ready, there's three things here that are not yet final, now you can take it when you, if you need it.

B: And that'd be an email, like, to the next person down the line?

I1: Down the line, yeah.

B: Or to the department, like a Shotgun-type thing?

I1: Oh yeah, it could be, it could be either or, yeah. So it could be down the line, or it could be to, um, let's say you're working on a shot or a sequence, it could be to that whole sequence mailing list, just so that everyone who's at least in that bubble is aware. I mean, you don't necessarily need to make the whole production aware, but there's always people, subsets of people working on different shots or different

sequences. And, um, yeah a lot of the times our emails will revolve around those sets of people.

B: Cool. So, yeah, I i guess, grouping, having people in different departments working on things, it makes it easier when you can, when you can tell them all at the same time, like, "hey this thing is ready."

I1: Right.

B: That makes sense. Um, you talked a little bit earlier about the need, like one of the top things for a pipeline being the flexibility, and the ability to adapt to new shows and new technologies and integrate those things so that you can use them for production. What, what areas of a pipeline need to be flexible?

I1: That's a good question. Um, fle–the, the generic answer to that is everything. [laughs] So I'm trying to think of ways to be more specific. Um, the areas that need to be flexible are, especially the, um, man, let me just think a little bit about this answer before I get, make it too confusing. Um, anywhere where you need to be able to integrate a new technology needs to be flexible. Um, you know, a new technology could be, uh, a package inside a third-party piece of software or could be a completely new third-party piece of software, kind of like, um, I know that, you know there's all types of packages for maya, and there's, you know, other software like HoudinI, things like that that other studios are constantly trying to, um, adopt, and so being able to be flexible in that way, that, if you're using, for example at Studio Alpha we use proprietary, um, file systems and file structures. Being able to do the whole export/import type of workflow, um, that type of stuff needs to be flexible, because if you're not, if you have your own proprietary file, files and file systems and file structures and you can't be flexible in adapting new technologies, then your'e screwed. You're basically stuck to what you've done for the last 20 or 30 years or whenever you, you know, started that, um, that proprietary stuff.

B: Right. So if you're, like say you're maybe, I don't know, writing out some animation data to a particular proprietary file format, you want to be able to plug that into different software packages?

I1: Exactly. You need to be able to, um, what we call condition the data from one package to another, in this case, from the third-party package to your proprietary system. So, that, that's basically, when I talk about flexibility that's the main thing I talk about. Um, uh, flexibility also in just, um, being able to not, um, how do I say...being able to not necessarily dictate how you want an artist to work; being able to, um, provide a solid foundation but also, you know, be like, your, observe how people work in whatever system they're working in or whatever workflow, and not, not being stuck to, you know, what you decided for them. They, they have to be able to inform you. It's a two-way street. Um, that was super-ineloquent. But, um, but...

B: [laughs] no, that's great and that, that kind of leads onto the next thing we're...

I1: It's kind of like, okay, so...if I can think of a good analogy, actually this will make it much clearer. So, you know, when you go into a grocer or supermarket, someone's engineered how the aisles go, you know? Someone's actually thought about, you know, "the milk goes here, the eggs go here, the cereal goes here." um, and they'll place it, and they'll, they'll be informed. They'll do some research, they'll do some social engineering research on what, what constitutes the best layout for this grocery store. Um, and then, as they open the grocery store, they'll watch and they'll observe how people move about the grocery, right? And so, for example, you know, if I put the milk over here, and I put the cereal on the other side of the, on the other side of the building, it forces people to walk through all the aisles, and maybe you'll sell something else, you know? Maybe something that they never, you know, intended to buy in the first place. Um, and as you watch them, you know, shop, you'll be able to

move things around, as, like, you know, given your observations. And that's kind of what I'm talking about with the pipeline. You have to be flexible enough to observe what you've already implemented and change it, and not, you know, get stuck in a place where you can't change anything, because, you know, then you're stuck. Then you're stuck with an obsolete pipeline that people are unhappy with.

B: And you want to keep moving people through the store, so to speak, you know?

I1: Exactly. Yeah.

B: Cool, that makes sense.

I1: Ok. [laughs]

B: No, that's good, that was a really, that was a great analogy actually, yeah.

I1: Good, thanks.

B: Awesome. Um, I guess, you...from that, um, you talk about s–, um, instead of just sort of handing...so you have the person who engineers the grocery store, and they decide where the aisles are, and how things are laid out. Who, who does that? Who decides, who's the engineer that decides, "the milk goes here, the cereal goes here?" like, is it one person, do they sort of hand it down from on high, or is it sort of a committee thing that...

I1: Um, it's definitely a committee thing. Um, in the case of, you know, our studio, we have a group of, you know, pipeline people that are, you know, heads of pipeline or supervisors of pipeline. Um, we have all the supervising TDs, um, also, and we have a head of TDs. And we, they all get together, and they discuss these things. Um, you know, I know that, you know, at the beginning of productions, major productions, one of the biggest things people need to come to a consensus with, um, and these are the consensus-driven decisions that you kind of allude to, um, are whatever the big-picture or big-target development things that they require

for that production, you know? So, um, let's say you needed a - I'm gonna speak from Big Film Two since I had most of my experience from there - but we had to, and this kind of goes back to your whole flexibility thing - um, Big Film One, our previous movie at Studio Alpha, we developed this whole city metropolitan system, to kinda build and, procedurally build a, um, a grid city system, and in Big Film Two it took place in Europe and European cities are not grid-based, so we needed a way to at least leverage off of some of the technology built on Big Film One, and this is the whole flexibility thing - we have to be able to pull that in, but also build upon it to make sure that, you know, the cities don't look like, you know, New York. They need to look like Paris, they need to look like Rome, they need to look, um, like London.

B: Yeah.

I1: And so, um, that was, that, that's kind of one of the things that the consensus-based drivens, or, consensus-driven decisions are done in the beginning of the show. One of the tools, for example, that we know from the beginning of production that we'll need is something to take that old city system and make it so that it looks more European, um, and in the case of the Big Film One world it looks more whacky - we call them 'whack passes,' you know - we have a, like a modeling facade type of, um, tool that, um, that took like a regular piece of geometry, like a square piece of geometry and whacked it up, and so that the proportions were all funky. And that's kind of what we do with Big Film One. Um, so those are the types of decisions that are made at the beginning of the film, and then later in the film, um, decisions are more reactive. Um, you've already developed the big systems that you need, like a level-of-detail system we did on Big Film Two, we did, you know, the modeling facade system that I told you about, and we did many systems, but towards the end, you know, or towards the middle, you'll start getting the, the decisions that

are based by, you know, what artists are encountering, kind of roadblocks they're encountering, um, things that they're requesting, because they're not as efficient as they could be. Um, I'm trying to think of an example. So we had, for example, a system for our layout team to propagate level-of-detail changes in the buildings. Um, we call them 'scatters,' and so they're, um, buildings that will change depending on, you know, how far they are from camera...

B: Right.

I1: But also they're art-directed. And a lot of people don't do level-of-detail art-direction. And so, that kind of stuff was...we anticipated in the beginning to change not as often as it actually ended up changing. And so, we needed to develop a system later on that prevented lighting renders from getting trounced upon by these propagated changes. So that's one of those pipeline decisions or developments that happened after the fact, and so those are kind of, two distinctions, in terms of who informs decisions.

B: Nice. And is there a lot of granularity as far as...you talked about those reactive decisions later in production, um, and when I say granularity I mean like farther down, like, to individuals saying, "hey I need x." you know, and then you just whip up something quickly for them.

I1: Yeah, because what I, what I'm talking about are the big big decisions...and, of course a huge part of the TD's role, especially at Studio Alpha - I mean, I'm sure everywhere else, a pipeline TD - um, are just the little tiny things that an artist requests. I mean, "hey this, um, the circular motion blur on this wheel looks funky, can you fix something?" um, and then you write up a quick script or you find a way to hack it such that it looks good according to the art director or something like that. Um, and those type of little tiny things are also, you know, they're informed decisions based on your artist that you're supporting. Um, and that happens all the time, and

those are like the very little one-offs that take, you know, maximum a week.

B: Right, right. Very cool. Um, I guess then, you know you've, you, like you mentioned you've been there for almost seven years. Um, how, how has the pipeline evolved over the course of your time there? What was it like at the beginning, um, what are the kinds of changes that you've had to make? Um, talk, just talk about those for a little bit.

I1: Ok. Um, I can talk about how much I can reveal.

B: Right, yes, of course.

I1: Which is, this one's the more sensitive one. But, um, you know, as I kind of said in the beginning, and I have some notes here, they're just, everyone's always trying to make their processes and workflows more efficient. I mean, and that job is never going to be 100% complete. It's always gonna change, it's just like how intel is constantly making smaller-nanometer processor, like, things like that, it's just always gonna evolve. Um, I do know that the pipeline has changed, uh, once majorly since I've been there, which is not bad. It's maybe averaging every three or four years. Um, and they're currently, um, and what I'm not allowed talk, to talk about is our next-gen pipeline, and that's coming soon. And so, what we did work on on Big Film Two was a new pipeline to me, as of two or three years ago, and it's, it was moving towards a more hierarchical scene graph type of, um, way of thinking about assets and shots. Um, and it was way, it, it was basically a new way of organizing shot data. Um, it was a better way of packaging, um, assets like geometry and rigs into higher-level types of entities, um, for the sole purpose of having more portability and better reuse of assets. Um, it was much harder to do those type of things in our older pipeline, um, and I think people felt that, and so this was dev–this type of pipeline, this scene graph-type pipeline, was developed by all of our pipeline engineers, um, that are brilliant, um, in order to make things way more abstracted, in a better way.

B: Right.

I1: Um, and make more flexible...and it, it was, it was meant to improve, just, the way artists work, because you were able to dictate, um, groups of things and assets in a given shot that you wanted to bring in to your environment or to your render. Um, previously you'd have to load everything, um, and so this was a way of, you know, one department can load these group of assets and change things. Another department can load another group of assets in the same shot, it doesn't have to be the same set as the previous department, um, they can make changes, um, and, and, and they won't hurt the upstream department, and they can propagate it to the downstream department. And those were kind of, um, it was a better way of, you know, it was a way of increasing complexity in the scene, but reducing the management that you needed in the scene.

B: Right.

I1: Um, and so that was kind of the new pipeline, as of two of three years ago. I believe it started with, um, Big Film Three, um, which was [came out a few years ago]. Um, and so the other major thing that changed in our pipeline was stereo—stereographic, um, rendering, um, because monsters vs aliens was our first, um, stereographic production, and so, it was, it was definitely a huge change because we're used to, you know, having deep files and, um, like, just, you know, one-eye renders, you know, just mono-eye renders, and now we have to render two different eyes, and, um, we have to figure, we had to figure out a way to rasterize our geometry, um, such that you're not wasting double the resources to rasterize the left eye geometry in space and the right eye geometry in space. We had to come up with a way, and these were r&d engineers, of rasterizing both eyes together, and then rendering out one camera versus another camera, which is another eye. And, I, I don't know the exact, exact numbers, but, it, it was definitely not double the cost.

B: Right.

I1: It was much less, and that was a huge change in our pipeline workflow. And so, those are, I mean those go back to all of your questions about how flexible you need to be. I mean, never in a million years probably ten, fifteen years ago did they ever expect to do 3D ever again, because they thought it was, I mean back in the day it was such a fad, um, and then, suddenly it became this technology that we could actually realistically do, um, and then we decided to, from that point on, make all of our films in 3D. So that was, that was a big decision and that required a huge, significant change in development of our pipeline.

B: Yeah, yeah.

I1: So those were the two biggest changes that occurred while I was at Studio Alpha. And now, we're constantly, I mean, like I said we're constantly trying to evolve, and so we're already thinking about not just our next-gen pipeline, but the next-next-gen pipeline, and so people are thinking up to five, ten years ahead, um, because I think the ultimate goal in, in feature animation production is being able to, like - it's kinda like gaming - being able to light and render in real time, and animate in real time. I mean, that, that can't happen with our current processing power, it's just not gonna happen, especially to the quality that, that feature animation requires, because, you know, it's a supremely iterative process. I mean, it's already an iterative process no matter where you go - gaming, animation, TV - but in feature animation everything, you know, it's blown out to, like, the big screen, so everything has to be pitch-perfect. And so, um, being able to animate and light in real time is the ultimate goal, and that's what these pipelines are trying to, um, ultimately achieve when they're making things more efficient.

B: Yeah, I, I just watched the new film trailer yesterday and...

I1: Oh cool, did you like it?

B: Yeah, it was gorgeous. It was really really cool. I, it's hard to imagine something like that running, running in real time, but I guess that's the goal.

I1: That, yep, that's the goal. Um, but other things I can talk about in terms of, um, what, I mean I can probably speak for everyone in the industry, just for future pipelines, you know, everyone's trying to improve their asset tracking, just like I mentioned before. Um, even being able to integrate it better, better in your environment. Like, right now, most people's asset tracking, or at least our asset tracking, is outside of our production space, it's just a separate piece of software.

B: Okay.

I1: Um, it would be awesome if, if, you know, artists could work, you know, in one area and see like, "oh here are the things I'm working on, let me launch them" or, "here are the things that are handed off to me or were handed off to me, and what I need to hand off." um, if this was all in like one, like if you can think of like, just iOS, like the iPad, and just one area where you can just click around and manipulate all the stuff that you need to do, um, as opposed to an operating system like linux or unix, where you have pieces everywhere and you have to navigate to them and it's a little bit more difficult. So that's something that, um, I'm sure everyone wants to achieve.

B: Yeah.

I1: Um, smarter assets is another thing that people want. Um, you should be able to, you know, I think when I first started we weren't able to just query an asset and be like, "tell me about yourself. What materials do you have?" what, I mean, this goes back, it goes back to like referencing, like how, what kind of, how deep you're referencing, you wanna be, um...just being able to ask an asset, like, "what kind of variants you have? What, where can I use you? Where are you being used right now? How long do you take to render in this lighting environment? How long do you take

to render in that type of environment?" being able to package all of that into one asset, in one smart asset, is something that I know people are trying to achieve.

B: Huh.

I1: Um, I'm sure people have, I mean we've achieved it to some level. And, so, I know that in my seven years at Studio Alpha, we've definitely gradually, um, made use of more metadata in files, and...so, metadata can just keep track of a lot of different things, like statistics of renders and things. I mean, we can, with metadata we can kind of predict, you know, how long a particular character is gonna take to render in a shot and kind of cost-project how much it's going to take or how long it's gonna take or how much it's gonna cost to render our a major character through 90 minutes of footage. And so that type of stuff you weren't able to do before, and that's another thing that pipeline, is pipeline related. So, um, yeah, that's, that's a, that's a big one.

B: Very cool.

I1: And, and actually, following through with that, you know, adding to that, like, the more smart your asset is, the better you're able to reproduce various things, um, in your shot. So if you're working in a shot or sequence, and you've got multiple iterations, and you've got smart assets, you can say "hey, I wanna render this shot with this version of this asset, this version of that character, this version of that, exactly how I rendered it six months ago." right now, I'm guaranteeing you that every studio will say to you that's almost impossible to do. You can't say to someone, "please render out this shot exactly how it looked five months ago," because everything's changed, you know? Like, everyone's assets are different, everyone's files are different. I mean, they are version-controlled, but it's almost impossible to, to gather...

B: To go back and grab them.

I1: Yeah, exactly. Especially since people's software, software maps are changing,

too. So, um, that's another goal of future pipelines is being able to have a smart enough pipeline that you can reproduce anything you want at any given time given a few variables or factors or specifications. And so that's a, that's a big one.

B: And I guess, in terms of...like for a director, that's probably really useful if they wanna...

I1: It's so useful, because can you imagine, like, you know, our new film has released its second trailer, it's not coming out [for a while]. Um, you can imagine like, later down the line, let's say they've iterated on some shots in the trailer and the director says, "hey I really wanna see that shot, the way it looked in the trailer. Like, can you get it for me?" like, that's something that, that's a very common request. Um, obviously trailer is a very definitive or defined milestone, so that's something that's probably easy to reproduce among all studios because they'll save that data somewhere. Um, but other types of very minimal-type things like, um, "oh, a dailies session last tuesday afternoon, like, please reproduce this." no one's gonna be like, sure i'll do that. It's gonna be a very challenging task, so. I'm gonna get more water.

B: Yeah, great. Oh, thanks. Okay, well you've talked a lot about the improvements that, you know, that are being dreamed up, and things like smarter assets, and, stuff to shoot for. Are there any, are there any things that you would personally like to see, that, that have, may, has or hasn't been discussed? Like anything that you, any particular tool or anything like that?

I1: No, I mean, all the stuff that I discussed are things I wanna see. Um, that, that type of stuff would make everyone's lives easier, plus it would make things just more fun because the, the benefit you get from a more, more efficient pipeline and a vastly, vastly improved pipeline like that, um, is that you get to do way more iterations. So you can make your movie look even better than it could have been before because you can do, let's say 50 iterations in the same amount of time you

were previously able to do 20 iterations. And so, that just makes everyone live, everyone's lives better, and so that's something I definitely want to see. Um, yeah, I mean that, there's nothing else I can probably add to that, so.

B: Cool. Alright, well, um...i guess that's the end of the questions I have written down. Um, is there anything that you, that maybe I, that I didn't ask about that I might have, um, I'm trying to remember if there was something I talked to you about maybe previously when you were in college station or something that...uh, was there anything else that you had?

I1: No, I, I've gone through all of my notes, which is great.

B: Good. Me too, I actually managed to fill up a page with some notes as well, so [laughs] this is good, it'll help, help guide when I transcript this thing and get it all printed out.

I1: Great, hopefully it'll guide, um, the, how you ask the questions to others, too.

B: Yeah, yeah. I, that, that'll be helpful as well. So, um...ok. I guess that's pretty much it then.

INTERVIEW 2

BRANDON: Ok, uh, this is the second interview today. I'm here with Interview 2 and uh, he's going to talk a little bit about pipeline stuff.

INTERVIEW 2: Hello.

B: [laughs] cool. Um, ok. Well the first thing, I'll ask about just so that we can sort of get an idea about your experience and the–the things that you've done. Talk a little bit about your background in the industry, um, what your position is, um, how long you've been there and kind of how you got–got into the industry.

I2: Ok, uh, so my most recent position is manager of project engineering at Studio Beta. And uh, my responsibilities are basically to manage central technology for all the different projects that we run there. We have two episodic TV shows, uh, feature film and we also support games. Um, so I have a team of about five people that are uh, engineers–software engineers–and we work very closely with the productions to figure out what their needs are.

B: Cool.

I2: Uh, as far as how I got into the industry, I started back in [the 90s] at Studio Delta in tech support. Um, they were expanding a lot back then, so uh, it wasn't easy to get your hands on 3D computer graphics software or hardware back then–It was very expensive. So I was happy to get in just anywhere and then be able to stay later and just kind of absorb and learn. Ah, and being in tech support, it was for pretty much everything–so everything from system admin kind of stuff to front–front–hotline support.

B: Taking calls?

I2: Yeah, it was like "uh, my Netscape is broken!" Um, I'll see what I can do.

B: [laughs] yeah.

I2: To questions about Alias and SoftImage that I'm like thumbing through the manuals and like figuring out like here they are–shouldn't you already know this? But I will figure this out.

B: Yeah.

I2: Um, graduated from that into software development. And um, wrote all kinds of 3D computer graphics sort of translator software. Um, pipeline software. Um, back at the time, they were transitioning from an old pipeline to a newer pipeline, so a lot of discussions and things with a group. Um, the last thing I did at Studio Delta was write, with a bunch of other people, and incline renderer. Because at the time it was integrate CG into 2D hand drawn films. Um, I wanted to get over into production. And not just be considered a, you know, programmer. And uh, so on the side, working with a few other people, we made some short films and learned how to model and rig, animate, write shaders, light and that became my demo reel and then I got into Studio X [later], uh, as a shading artist on Adventure Film One because they had to do the whole film in about nine months. [laughs]

B: Yeah, because of the–they had like a big crash or something–didn't somebody remove–

I2: Uh, that was–that was not why they had to finish the whole movie so quickly– It was because they retooled the whole story–like they had a different team working on that direct to video project and then we–yeah, and then [director] finished Adventure Film Five and took a look and said, you know, I think we want to make this a lot better. And then he took over and basically re-did the whole story. And so production was yeah, less than a year to do everything, um, which is pretty crazy.

B: Um, well I mean it turned out great, obviously.

I2: Yeah, and it was an awesome time. It was like 300 or 400 people at Studio X at the time–still in [location], so it was very small, relatively small, uh, compared

to how it is now [laughs].

B: Yeah, got–I actually haven't been up there, but hopefully I–I'll be able to go visit it maybe. At least swing by the building and see what it looks like.

I2: Yeah, um. So, I ended up being sort of the last shader writer on the show just supporting everything, um, because I guess I–I did well, uh, so then they let me do some sort of r&d in-between shows and then on Adventure Film Three, I got to do character shaders for like the main characters. Um, [character] and [character] and a bunch of the secondary characters. And then I said I'd like to light...I haven't done that before. And they said sure. Uh, so I got to light shots on Adventure Film Three ah, after that, I went into r&d for a while. It's a long...we're covering a long period of time here.

B: No, that's awesome.

I2: Um, so basically, we're getting ready for Adventure Film Four which is a very different kind of show than they'd done there before. It was humans, it was a lot more effects, a lot more sim, a different style of storytelling–a lot of shows up until then were, you know, a few locations, um, and you just kind of kept coming back to those main locations. This show had tons of locations, it had montage sequences, it had big action sequences, it had cities... yeah, and so we were uh, concerned about how to pull that off with the existing pipelines, and so a lot of what I did was figure out how to pull off Adventure Film Four on the budget that they had, with the people that they had, so I was uh, pre-production, uh–I forget what they called me in the credits–but at the time it was more like sequence supervisor. So anything that was not characters was my responsibility. So um, modeling, lighting, um, set dressing, layout, all that stuff up front was uh, my responsibility. And we ended up looking at alternative ways of doing stuff other than take the main pipeline. What if we had these little, sort of rebel unit kinds of teams, um, that could do stuff with more off

the shelf and just be nimble and uh, so that was a big experience there. I also worked in a more sequence oriented manner. Um, lots of changes to sort of workflows as well as technology.

B: Yeah, yeah.

I2: Um, after that, I ended up going sort of freelance for a while. Um, ended up with a few other ex-Studio X kind of guys and we ran a small little boutique CG shop, did consulting for other people. Tried to get some creative stuff going, too, which is really difficult. Um, and then we ended up–our biggest client was like the Orphanage, so we did a bunch of r&d for them and ran shots and stuff like that. Uh, it was a little too up and down being independent, so then I went back to Studio Delta as a–a...

[car noise]

I2: Wow. Pulling right up to the shop–that works. Uh, as a CG supervisor for Adventure Film Two. Uh, because this was when they were basically starting a whole new studio to basically do the [sequels], so it was starting a studio up from scratch with a bunch of people that had been at Studio Delta for a while and also a lot of people who we had to recruit from other–other places, so that was a real melting pot of ideas and um, we got to basically look at the Studio Delta pipeline at the time and say can we use that to produce the level of imagery that they want. Um, at the price point and with the staffing level that we have. At the time, the answer was no, we don't think so. Uh, it took a lot more people to maintain the pipeline than we would have had. So, we ended up building some pipeline stuff from scratch and running into a lot of opposition of course [laughs].

B: Yeah, yeah. I bet.

I2: So... yeah. Um, then they bought Studio X and they shut us down, so a bunch of the core team of us got absorbed back into the main body of Studio X

and we worked on other shows, but kept sort of tinkering with our pipeline in the background and eventually got to do a short and then the Adventure Series Two TV shows. And uh, now it's basically sort of–you know we branched and merged back together and so the pipeline now at Studio Delta is now sort of some of the really nice tools that they have, but also a lot more of the back-end stuff from the other pipeline. Um, and uh, and then I came to Studio Beta. They were starting up an animation studio to do feature films, and so here we are. Basically the same thing [laughs]. There you go.

B: Very cool. That's awesome. Well y-you clearly have a lot of experience in it sounds like a lot of different areas, too. That's really great and that'll hopefully provide some good perspective. So, based on–well, how long have you been at–at OverStudio now?

I2: Uh, so I'm not technically at OverStudio. Like OverStudio is a bunch of different divisions.

B: Okay, so you hover between them kind of.

I2: Eh, animation is its own thing–we use a lot of technology...so, one of our shows is using basically the OverStudio pipe, so I go back and forth a lot for meetings and coordination and stuff like that, but yeah, if you had to go technically, I'm employed by Studio Beta, because it's like... but anyway, I've been there for two and a half years–almost three years now.

B: Okay. Cool. Well in, talking from your–from your experience from all these different places and all the things that you've done, how would you define a pipeline?

I2: Ah, I guess, for me, it feels like there are a lot of different definitions for pipeline, depending on who you talk to and how they think about the process. Um, so that's why actually I drew this thing while I was waiting.

B: Oh, awesome. Interview 2 has drawn a Venn diagram that might even well–

some version of it might make it into the documentation for the thesis paper. Uh, but this is–this is good to look at. So go ahead and talk about it.

I2: Okay, so in this bubble, we've got production management, so they–they run the show, organize the teams, figure out how, uh, the work is going to get done. You have the creative which is art, animation, you know, the people who are doing the story who eventually what's going to show up on screen–we've got technology that's supporting that. Right in the middle is pipeline, right? It's figuring you how to use technology to support the policies that they have, how many people you'll have, what kind of people you'll have and then you're trying to solve the creative problem as well. Like I said, on Adventure Film Four, oh, we're going to do humans for the first time, so we need to have process tools, and uh, workflows that support getting that on the screen. So, it's uh, sort of where all those things come together–that's what I think of as pipeline.

B: Awesome. Well, that's–that's really helpful to look at, actually.

I2: A lot of people think that, you know, that some of these parts are the pipeline, uh, because that's what they're exposed to–like if you talk to an artist, and say what's the pipeline? They'll talk a lot about their tool–they'll say well, my lighting tool does this and here's this is how I get my shot. It's like okay, it touches on some of the pipeline stuff, but a lot of what they're talking about is just sort of their daily workflow. Um, I think the other thing to me about pipeline–It's really talking about how you share work with other people, right? Because if you're just doing it yourself, sitting in Maya or Max or something, and you're just modeling, rigging, lighting, I guess technically you could have a pipeline, but because you're not really sharing work with anybody, um, you might just be working in one file, it's just a workflow. Um, so, I think for me, the pipeline comes in when you really have to start thinking about scaling it up and involving more than one person, department...

B: Cool. That's–yeah, that's great because I know when I started, um, thinking about this thesis, and–and, I've built, um, basically like a set of directories and tools, um, for–I did it for our summer course last year. You get sort of trapped into thinking about the pipeline as being here's the directories where everything goes and here is um, the set of commands that you use to move data along the pipeline. But it's–it's more than just that.

I2: Yeah, I think so–because if I talk to a production person, they tend to think of the pipeline as being the departmental workflow. It's like well modeling goes first, then rigging, then they tend to think of it very linearly. It's like hey–here's our different departments and I have a deliverable and uh, you know I have some iterative loop here and then when it's all approved and moved down to the next thing and so they–you even see some pipeline software out there that's like–build your workflow with nodes, and I'm like augh! Because it doesn't work that way. Because, you know, modeling and rigging it's like uh, well, I can model and I can refine it. I can get approval. But if I send it to rigging and they send it back because my OSA params are in the wrong place... or uh, the legs are too short and he walks funny because we haven't put a rig in it yet, um, then you get into loops like this, right? Where it's going back to modeling, more iterations and then, so it can get confusing–especially the more departments you have that are woking in parallel. Like lighting and rigging could work–I mean–like surfacing–the shaders could work in parallel, and so now you have these multiple loops... and I need to send that there because I want to see if textures are going to stretch and when it's deforming and so now it's like this spaghetti. If you're trying to track every single dependency explicitly, it gets really hard. Um, and you have to expend a lot of energy and engineering and told to actually manage all of that.

B: Mmhmm.

I2: And then the poor artist is like, I...I got a new update? [laughs] do I use it? Do I not? And so it can be challenging.

B: Yeah. Cool. So what–what do you think makes–what do you think are good characteristics of a pipeline? What makes a pipeline good?

I2: [sighs] uh, my best analogy should be like your heartbeat and your breathing. Most of the time, you don't even think about it, right? Heartbeat you don't even control, really. It just sort of reacts to what you're consciously doing. It's supporting you. Um, your breathing–most of the time, you don't think about it. But if something's going on like a fire and you need to hold your breath or you're going underwater, you can control it. But most of the time, you're not even consciously aware of it. That, to me, is the best pipeline. Right, it's just out of your way–not noisy, um, and lets you focus on the creative work because that's really what we're all trying to do.

B: Yeah, yeah. Cool. That's a–that's a really good analogy. Interview 1, the last interview talked about the pipeline being sort of like a grocery store, so we're getting all kinds of fun–fun analogies about what a pipeline is. That will make for good–good work I think.

I2: [laughs]

B: So you mentioned, when you were drawing your diagram, touched a little bit about feedback loops, so, that kind of stuff, like you say, can spiderweb and get complicated kinda quickly. What kind of communications tools do you use to keep track of that, to manage it, to make it easier for people to know what's going on?

I2: Um, [sighs] well, a lot of people like email.

B: Yeah?

I2: Uh, I'm–I have mixed feelings about email. [laughs] Because it's inherently noisy, and uh, if I'm an artist and I'm getting an email every time there's some update

to an asset in my shot, I can be getting lots of emails every day, every time somebody checks something in–and most of the time, that's just noise. Um, I don't care. When I really care is if something broke–then I want to see well, what was the last thing that changed? Um, so, you know databases would be better–with easy ways to query. Uh, RSS feeds, so it's like, it's all there and I can go back to it, but not...

B: Necessarily showing up in your inbox automatically.

I2: Yeah, yeah. Um, [sighs] yeah, I don't know. I guess it depends on the context. There's lots of different kinds of communication that goes on. Like if I just want to see notes for my shot, well that's review. Well then chances are I want to go to my production tracking database tool, um, and pull up a page or something, my particular information–you know, filter stuff. So I guess for information, I tend to think of for the day to day stuff, let the user–end user pull and filter how they want to see stuff. And the only time you push information to them is when it's like you really need to know this, or yeah, something bad is going to happen.

B: Right, yeah.

I2: So it's like oh, the rig's changed. And they're not backwards compatible. Then, we might notify everybody, versus, um, the rig has been updated and that weird crease in the elbow is not there anymore. It's like I don't care about that. Does it change the way I animate? No... so, yeah.

B: Cool.

I2: And then there's just coordinators walking around actually checking in with people, um, that... more than any tools, I think, makes a difference in how smoothly production can run.

B: So those are–those are just people who are supervising or?

I2: Um, they're not necessarily even supervisors. They're–they work with the department manager or the production manager and they have sort of the status of

everything sort of in their head. And they might do some data entry later and make sure it's in the database, but they'll check in on people and see how they're doing. It's like so, you know, are your shots going to be ready for review this afternoon? Something like that you need to review. Emails and other things like that really, uh, it's just actually faster and easier to have somebody with a clipboard going around gathering information and then making sure that everything happens. I guess–that's what I've found.

B: Yeah.

I2: I know other people have ways of like oh, go to this page or type in this thing and submit a shot to dailies and it'll automatically make a playlist and that stuff. And that's stuff I see more of as an aid to the person who has to actually like run the dailies and like go around and collect all that information, because if they're just looking at the list of stuff after people have submitted it, they're missing context. They're missing if I stop by and ask what's going on, they say, "well, I'm going to show this, but these things are broken or I only need feedback on this" so... and unless they're taking extra time to put those notes in–and some systems don't even allow that, you know, then it's–It's a mechanical help. It solves you–It saves you from some extra data entry. But, actually making sure that the right things get put in context for the person doing reviews, um, or–you know, prodding people to say, "you know what? That's good enough to review. You don't really need to wait another day." That kind of stuff. You're not really going to get somebody actually walking around talking to people.

B: Yeah, that's cool. So there's–there's more of a–more of a human element that makes communication easier.

I2: Yep.

B: That's under that production management circle in the Venn diagram you

drew. That's very cool. That was um, I mean, Interview 1 talked a little bit about that, um, from the last, but there was much more emphasis on that from what you've been saying. That's cool to get that different perspective.

I2: Sometimes I think it's because artists are, sometimes introverted, and so they're like oh, if I can just type it in, I don't have to talk to somebody and that should be, um [laughs] easier. I know we had it, in some cases... um, there was actually, both at Studio Beta and Studio Delta, they had a, um, playing your dailies' playlist and as you go through, it's like IM-ing somebody or broadcasting like "Hey, your shot is coming up next. Please come into the room."

B: Yeah, yeah. They have something like that at Studio Delta–I've seen it. It's like you're on deck, you know.

I2: Yep.

B: You're coming up next.

I2: Yeah, which is nice, but I kind of miss–when it was like a Studio X when it was smaller and you were just all in the room, and hearing everyone's notes for shots that weren't yours, but were related and you can sort of head off things and see what the director is getting at–Instead of just getting your notes–or the person before you and after you's notes...

B: So it's a little less self contained?

I2: Yeah, because a lot of what we do for the film is about context. It's what is the director or the art director trying to achieve? And if they have to repeat themselves, you know, for every single artist, somethings just get lost because they get tired and they won't say the exactly the same thing to everybody–

B: Right.

I2: If they say it once to everyone, then I think it helps foster the notion of the team. It helps everybody, um, sort of get on the same page, um, um, and really

understand what's important. Because you can noodle anything in a shot for–forever. So sometimes it really does help to be all in a room and hearing notes at the same time.

B: Cool. Um, you mentioned a little bit about–talking a little bit about the communication and feedback loops, um, and a–a need for version tracking and asset management. Um, what are different methods that you've come across in your time for the version tracking or what are–what do you think is I guess an effective way to do that? I know that's kind of a global question, but...just talk about version tracking and how important it is.

I2: [laughs] um, that's my other diagram.

B: Alright, sweet! I'm hitting all the diagram questions here.

I2: So... I–this is sort of data management as a stack. Uh, and it's not a purely technical stack. I dunno have you done any network like protocol kind of...?

B: I did in one of my classes.

I2: So they have like a network protocol stack...

B: Yeah, yeah, yeah.

I2: At the bottom's a physical layers...Yeah, so it's a–this is loosely inspired by that.

B: Okay, okay.

I2: Except for the top part–there's no tag. What this is is policy. It is what you decide are the rules of the road for your show. Um, so, up here could be something as simple as–have you heard the terms push, pull?

B: Yes.

I2: Yeah, so up here, just say well our policy for this piece of data or department is push, but for this department, it's pull. Um, so... that is more of a people and organizational kind of thing. Below that, then you actually have your user tools. So,

um, RenderMan, Maya, whatever... and those are going to talk through some sort of pipeline API just to buffer you from having to know, what's your file system? Is it Linux or Windows? Ideally you have just something that is agnostic.

B: Right.

I2: Under that, you have asset management which is where you get to more of the pipe–technical pipeline stuff. To me, asset management is sort of like software configuration management. It's saying for this particular circumstance or situation, what versions of everything do I want to combine together into these things that I'm actually going to use. Right, so I'm, um, going to need version five of the character with version two of the props and version six of the lighting and so ah, all that combinatorial stuff, uh, is asset management and it's pretty–can be tightly coupled with version tracking, but you don't always want it to be the same thing. Um, because production tracking status of like approved or not really doesn't have very much to do with data. Um, it might say which version is the approved one, uh, so that I can verify that I've got the right version. But if I combine the systems, then I've got like all of a sudden uh–what if I want to use a different–uh, unapproved version because I'm testing or other types of things, so you actually have to be careful how you do pipeline stuff.

B: Mmhmm.

I2: Under that layer, then you have revision control. And this is, um, ideally content agnostic. You don't care what is in the files, you just want to say, um, I've got a blob of data and I want to revision control it, so just destroy the version history, logs, be able to check-in, check-out. Um, very classic software revision controlling. And the very bottom, we've got tech. Uh, which is just the hardware, um, your os–Linux or Windows, or Mac OS–whatever you want to use. Uh, what kind of file systems am I using? What network and I using? Uh, am I distributing stuff between

sites or is it just internal? Um, all that stuff is sort of at the very, very bottom. So revision control–It's super important because, um, you always get asked to go back, [laughs] um, but you have to also be careful about guaranteeing reproducibility because a lot of systems jump through a lot of hoops to say I want to be able to go back three weeks to that shot and get exactly what I got out of it before–just hit render and–It sounds like a good idea, but I think the return on investment, based on how much engineering has to go into guaranteeing that–uh, it's not really worth it.

B: Really?

I2: At least not to me in the end. Because I can actually have a good artist go back to a shot, pull the data, and if it doesn't work right off the bat, they can fix it. I mean, they know what they're going for... I think the–the big issue is if you want to go to the director and say it's not exactly the same, and then they're like, "No! I want it to be exactly the same!" so a lot of it–that comes back to policy. It's well, do you really need to keep it exactly the same or it's okay if if it's mostly the same because you're going to give more notes anyway. And we could spend three man-years engineering something, or we could spend three months engineering something and focus on building tools that make your pictures look better. Plus, you have, uh, stuff that's not even part of your movie that's part of your equation like–uh, what version of Maya was it? You know, and we patched everything. Who knows? You can have so many different variables in there that yeah, to me is just... in some ways it's like somebody asking–I want to control every beat of my heart.

B: [laughing]

I2: It's like do you really want to or would you rather just, you know, jog your lap or whatever you're trying to do, so...it's sometimes very hard to come up with the right ways to explain to people–you really don't want that. It just sounds like it

would be safe. Was that, uh...along the lines of...I think I drifted a little bit.

B: No, that's absolutely great because, you know, actually one of the things that I discussed with Interview 1 was that–that ability to go back and be able to find something exactly the way that the director wants it from, you know–like say they're using something from a trailer, and they're like we want this shot exactly like it was in the trailer even though it might have been iterated a couple of times since then. Um, so that's–It's good to hear sort of a different take on the–on the utility of that.

I2: Yeah, I think that it–It does come back to the people equation...and, the kind of project that you're on because if you're working for a client you want to guarantee that I can go back exactly because I need to make the client happy, verses um, at Studio X or Studio Delta where the director is internal, um, you're iterating on the whole film over a long period of time. It's like well, maybe you don't have to come back to it exactly, because when you come back to it, you actually want the latest, greatest thing because it has fixes in it, uh, they might have changed some of the creative, so it's like well, it would be nice if I didn't have to manually update every single shot.

B: Yeah.

I2: Um, which is one of the reasons I tend to like push–managed push, not just everything updating all the time. But um, it really saved us a lot of heartache. Like on Adventure Series Two, we had finished animating the first sequence of the show, and they decided they wanted to change one of the characters.

B: Oh no.

I2: Um, so we were able to um, update that one character's model and rig and leave the animation there and just re-render the whole sequence, uh, without having to manually touch every shot. We just said, you know, give us a list of all the sequences, fire it off on the farm. Comes back, look at the movies and say oh,

these five shots need to be fixed–the rest are okay. Um, so that kind of flexibility and avoiding the need for manual intervention, I think for me is–that's what you want a pipeline to help you with.

B: Mmhmm.

I2: Right? You don't want to say, "well, I have to go touch now 50 shots in this sequence and I'll write a script or something to go update everything."

B: [laughs] yeah.

I2: It's like well, technically that is pipeline and the end result is the same, but, you know, I have to now pay somebody to write a script, they have to babysit it, they have to check to if they did it right, verses if the pipeline has already taken care of it through referencing or whatever, it's like oh–I just update the released thing and go. So, uh, yeah. A little bit of preaching here. [laughs]

B: [laughs] That's okay–that's what I–that's what I want. I want you to preach. I want you to get on your soapbox and–and–talk about what should–what should be there and what you have found is good and effective, so...

I2: Um, the other thing you were saying–what's the sign of a good pipeline?

B: Yeah.

I2: Have you looked at Unix philosophies? At all?

B: Um, briefly, yeah.

I2: Mk, because that was one of the things that, when we were collecting ideas, and trying to convince in the face of a lot of opposition, was well these are a lot of good things that make sense for pipelines. Um, so I think if you like Google online for Unix philosophies and Eric Raymond, you can come up with–I forget how many rules there are, I made a little note, rather than misquote, I will look it up. So it's a lot of things that are just nice engineering principles in general...uh, it's not downloading. [laughs] But it's things like modularity. Um, build simple programs that can do one

thing really well, but can talk to other programs and um, basically if you have to fail, fail early, fail noisily. Yeah, rather than hiding problems, um, well, that's a shame. [laughs]

B: It's not downloading?

I2: No. [laughs]

B: Well, I'll definitely–I'll definitely go look that up and look at those later on. That could be useful.

I2: I can also send you a link later on.

B: Yeah, that'd be great.

I2: Um, but we... at the time, we were reacting to a lot of systems in our mutual histories from different studios that were very monolithic, that uh, it's like well I like the revision control, but it's tied in the production tracking, so now I can't use a different, you know, way of tracking, or revision control. It's like well, we want to update from CVS to Subversion or something and it's like oh we can't because it's tied into our tracking database. Right, so building things to be modular, I think–also at the time, we were really just trying to make things small, simple, um, simple not necessarily meaning easy, but pared down to the basics, just what you need to get the job done, avoiding a lot of extra complexity.

B: Yeah.

I2: Um, but yeah. Okay, I lost my... [laughs]

B: [laughs] no yeah, that's–that's great. Well, you, um, you talked a little bit about flexibility and you talked about the transition, you know, at–at–Studio X like having to handle new types of–of, uh, character needs and technology and things like that. Um, I guess talk a little bit about, uh, flexibility in a pipeline and what kinds of things it needs to be able to handle. Like how–how flexible can a pipeline be before you just need to tear it down and build a new thing?

I2: Hmm. [sighs] well, that's actually coming back to Unix philosophy. So if you have modular pieces, then they're fairly easy to swap out. Um, if you have well defined interfaces between the different modules, then you can replace it with something because now you know how to talk to the rest of the pipeline. Or else what's happening to the other side of that interface. Um, another one of the Unix philosophies is build for the future, so um, make sure that, uh, you don't hardcode business logic into your code, but uh, the code is actually pretty stupid and it's just reading config files that are telling it what to do, and that sort of thing. Um, yeah. So a lot of the really strong pipelines that I've worked with tend to use metadata to describe their assets. So, like a character–you've got, I don't know, a Maya file for instance, but that's not necessarily the way you build your shots. You don't just take Maya files and reference them directly. You come up with a little text file that is easy to read, easy to change and says ah, for this shot, these three files belong in there, and it's pointing at those files. Ah, and then you can get fancier and say well, every asset has different representations for different packages that we use, so there's a uh, you know, a Houdini format or a Maya format, so I–I can read that same scene description and go into any app and load up my assets and shots and still do whatever work I need to do, and then update that metadata with whatever I've added back into the equation and now anybody else can go back into that–that little recipe and pull stuff out.

B: Yeah, yeah.

I2: It also allows you to, um, work with things that wouldn't fit into your machine's memory. Like back in–on Adventure Film Three, I think on the scare floor, we talked about having tens of thousands of assets because everything down to the push pins on the desk for the bulletin board, and there was like 12 stations and every piece of paper is its own asset, you know, these are complicated things that

actually, if you tried to load the whole thing, it would kill your machine. But by having that little, sort of metadata way of saying well this is all ten thousand things in the shot–that fits into memory. And I can say, "well, what's important to me today is the character in the floor because I'm animating him and that's all I need." right? And RenderMan can handle a lot more, so you need, you can say, feed it the whole description and it can load in the geometry and process it and give you out the complex picture. And that was one of the striking things to me–going from Studio X to other places–was that they were talking about a complex shot being like 50-100 assets. I was like what?! This is killing your machine? [laughs]

B: [laughs] Yeah.

I2: Right? And then looking at how they put it together, it's like, okay, I see why. Um, because they didn't do that sort of abstraction where as, I think, like Studio X and Studio Alpha and people who had built their own pipelines in the late 80s and early 90s, uh, had to do something like that because otherwise, they could never actually do the work they were asked to do. Um, and so that was really striking to me. I'm still wrestling with things like that–of direct representation versus levels of epic–of abstraction. Um, because a lot of pipelines are built by artists who happen to know how to uh, you know, sling a little code. Not necessarily people who are software engineers. Um, yeah, so I think [sighs] my experience has given me a little bit of interesting perspective that not a ton of people have–I mean, there's definitely more than there used to be.

B: Right, right.

I2: Um, but, being in a software department, it's like okay, how do you engineer stuff? What's possible with computers? Uh, what are good practices? Uh, being an artist, you say well, you know, I'm–I'm being asked to turn this around in, you know–I get my note in the morning. I need to turn this around by afternoon because

I have another review. So what's important? What's going to be in my way? What's going to make my life easier? And then being a manager and a supervisor and saying well, for the artist, this is kind of an annoyance, or, it seems like it would suck, but on the global scale for the show, it actually makes running the show easier. Um, so...yeah. Bringing all those perspectives to the table, it's one of the reasons why I had to come up with this kind of thing–because it was...a lot of times, you optimize for one thing...or maybe–maybe two things if you're lucky, but trying to find that balance of everything working in harmony and being able to make the trade-offs like oh yes, yes your work is a little bit harder in this case, but it's easier for you to get data from the person before you. So, you know, is that worth it?

B: Yeah. Awesome. Yeah–yeah. That's great I think. It's is unique to come from all three of those areas, um, because, um, most of what I talked to Interview 1 about was probably more tech related, uh, because that's kind of what TDs at Studio Alpha do is tech related stuff. So, um, that's really cool– it kind of expands the definition.

I2: One of the other things that we... in talking about flexibility. Remember I talked about people making these node graphs of like hey–and then data flows from a to b and all that stuff. Uh, the thing that we kind of looked at was more of a software thing–paradigm where this is the show. And in the show, you might have a shot or you might have an asset, um, and those are comprised of even smaller pieces– individual files. But, this is the show, um, and rather than say you have this linear thing, you take it out, you work on it in your sandbox where you can make changes without affecting anybody else and then you check it back in. And somebody else working out here, if they don't have the particular piece that, um, they are editing, then it just pulls from like a global pool. So it's more like, um, you sample the data that you need from this ever-changing mass of stuff.

B: Yeah.

I2: And it's super flexible because now it's like oh, we added a new department in the node-to-node thing is like well, add a new department in the ins and outs, define the way you can do something. And it's like well, what if the software department is busy or the TD is busy and I just want to test–I don't know what I need yet. Um, this allows you to test those workflows and ideas, um, without having to change the entire pipeline. Because it really is just like hey, there's a defined slot for stuff, you can define those on the fly because you're using metadata or something. Um, and you can do local testing without having to globally go to the core tools and update them. So anyway, it's a very different, once again, philosophy.

B: Yeah, and the diagram that he's drawn is sort of like a–It's a circle with some like you said assets and shots in it and then you have individuals outside who sort of check things in and out and, uh, and back into those predefined slots like you said, so...

I2: But this is pretty much classic SVN, Recurial...you name it. That's pretty much how they mostly think. Uh, and it's just trying to take that paradigm and get artists use to it because it is–It feels very foreign. Um, especially when you say to somebody, uh, the file is named foo and no matter how many versions you make of that file, it is not foo under bar one, because the revision control system takes care of all the versioning for you. You can have notes for what the revisions are, uh, but you still have people that don't trust the revision control system, uh, and sometimes for good reason because whatever studio build a custom one, uh, and it's database and all of a sudden the database goes ah I can't get to my file! [laughs] it's a good thing I kept foo number two in my home directory! Um, so, I think, that's kind of like this legacy because of the way things have evolved is not necessarily the way things are today, especially with more modern revision control systems. So I think, that's

something that I can still see–can still see some evolution happening there–at least at the places I've been.

B: Yeah, yeah. Um, yeah, I kinda know what you're talking about because last–last summer, we had, you know, some published commands and things for shots to be done and people were saving like three or four different versions of a shot as they were working on it and we have a–you know a snapshot backup running in the background all the time anyway. And so they were saving like eight versions of the same shot and then when they published it, they published shot.ma and the one they wanted to publish was shot06.ma and so it screwed up the stuff down the line because they had forgotten to rename something and so, yeah. I can speak to the level of mistrust of, uh, systems that are in place. Um, you talked about...so, you touched on the production management sphere a little bit, and how they come up with policies, right? That's the production management branch that sort of comes up with a policy? Or...

I2: Um, sort of. Right. Says you know, I guess at the top of the heap would be sort of the... the producer, the executive producer on the show and they'll say well we have this much money, and we have this much time, and then it's sort of the production manager and the visual effects supe and the CG supe's responsibility to go figure out well, I have this constraint now–how do I actually organize the people to go do that? So, in the, more successful collaborations that I've had, it's usually been department supervisor along with like a technical person or artistic kind of person that's experienced in that discipline, and they're uh, department manager who is more uh, I guess more of a management type position. Uh, and they collaborate on figuring out the best way to meet the time and staffing and results they're supposed to get. Uh, on bigger shows, you'll also have a technical supervisor–like the tech–like Interview 4 is a tech supervisor. Um, and he'll work with all department heads to

figure out um, is that really the best way to go? Have you considered that they have to do this other thing and then so... and balance all the department tool designers and eventually come up with sort of a way for the show to go, but, uh... I don't know how specific you want to get.

B: No, I mean, I–I guess the question–you know, you talk about the decisions of how–how data is structured, what metadata is attached to things, how assets flow through um, or are checked out in the pipeline, and I was just kind of wondering who–who decides these things–who says–this is–we're going to use a push system and here are the–here's the set of software we're going to use for, you know, who makes those decisions and is it sort of like this committee of–committee of people or...

I2: Uh, a lot of times, it's already established, right? Because you're at a studio that's been running for a while, so... there's already departmental procedures and things in place and you're not going to change those too much. So, I'd say in most cases, it's already decided because it's been done before, and it's not worth it to go redo it. Um, I've been, for better or worse, in the position of trying to have to challenge some of those assumptions, um, and that's where it's really tricky. It is sort of–It's not necessarily a committee, uh, but there's a lot of discussion. And [sighs] you kinda need buy-in at a lot of different levels, right? You need the supervisor's lev–like the visual effects supervisor's buy-in, the tech supe's buy-in, um, the individual department leads have to buy in, or else it doesn't have a great shot of working because they're not invested in making the show, but things move...then it's just tough. Um, so, yeah, not necessarily a committee, but certainly a lot of discussion between different parties and who would get affected and who would have to do the work. Yeah, when we were talking about push-pull, it was like everybody–I mean, it was like every supe in the building [laughs]

B: [laughs]

I2: Right? Uh, before I came to Studio Delta, they had just finished, uh, a long series of meetings on naming conventions and directory structure. So...yeah, I guess maybe not officially a committee, necessarily, but a lot of people trying to figure out what's best. And one of the problems is there's no one right answer, and so, some of it just gets into [sighs] personal preference–this is what I'm familiar with–and some of it is what are you trying to optimize for? I'm trying to optimize for readability or I'm trying to optimize for parsing, you know, um, for an automated thing to go. It's like–It's all over the place. Yeah, I–I wish there were one right answer. But I think that's why when we were talking earlier about being flexible and not making decisions in your code, but actually coming up with, uh, a configuration kind of method,

B: Right.

I2: That then your code can just execute, then that allows you to be flexible. So if you have, like at Studio Beta, we have episodic shows, so we have seasons and episodes, then sequences and shots. But on a feature film, you just have sequences and shots, maybe an act if you really wanna do–you know, look at that level. Um, but then as soon as you talk about directory structure, it's like well, I have a season episode, uh, sequence and then shot, so for some people to then go say, "I will"–do you know Python?

B: Yeah.

I2: So ospath.split–fifth thing in is the shot. Well, I can't use that code on my feature. But if you said give me the shot name and use this API, then it doesn't matter what the directory structure is and your code is reusable. So it's that kind of thing, so... it's really easy to go ah, ospath.split because then I'm done, versus looking up the API and, you know, I've got my supervisor going uh how come that script isn't done yet? [laughs] then you take a short cut and it works and nobody cares because it's like well, the shot got through. Or you fixed that problem. But then we

come to the next show and it's like oh–you now I remember this thing we used on the last show and now I want to use it on this show and it's like ok, we can't. We have to write it again. So, those costs are hidden. So a lot of people don't account for them, but they do add up. And for pipelines that have been running for a while without a lot of oversight, like technical oversight, um, you can get in a pretty awkward place where the maintenance costs of doing stuff or incremental improvements of things take forever. And uh, you know, it hampers your ability to be, especially in this industry, nimble and agile and do things that nobody else is doing because you're spending time writing things to parse directory structure instead of writing, you know, a new simulator or something like that.

B: Right. Yeah. And I guess the last–the last thing that we–the last thing I have written down anyway is, uh, just evolution–you've been–you've been in industry for a while, um, how–there might not have been any kind of see change or anything, but how have pipelines at different places evolved over the period that you've been in the industry? And I know it'll be different in every studio, but, um, have you noticed any trend of evolution as you've been at different places?

I2: Um. Hmm. Hard to say. I guess on pattern that I've noticed is that everybody would like to be doing stuff other than pipeline. And so... pretty much every couple of years, we go investigate some off the shelf solution, and find out that, okay, it–It demos really well, but it doesn't hold up for the specific work flows, um, and tailoring it to our specific studio work flows would almost be like writing another pipeline. But it's just–you're using a, uh, third party framework. But then, now you've got challenges there because now you're like eh I could really use this change in the API, but they're not going to want to change it verses where if it was in house, you could, um, and sometimes they just don't scale very well–they're not–you know, they've been tested at studios that are like a medium sized studio, and when you try

to throw a Studio Delta sized project at it, or a Studio Beta sized project at it, it's like ah–It just chokes! So...um, but we keep coming back in the hopes that someone's crafted... [laughs]

B: [laughs] in the hopes that someone's going to figure it out. Yeah.

I2: Um, I've also seen it go back and forth between, well I guess it's more of an evolution, so back in the early days of the pipeline, a lot of stuff was based on just the file system, and disks and being able to, uh, find things because it was pretty robust. Problem is, it didn't necessarily scale very well because, you know, you have your disk servers being hammered by these, the shows got bigger, render farms got bigger and all of a sudden, these poor disk filers are just dying. So a lot of stuff has moved more to databases. Um, which is nice, but then it also has happened where the database goes down, and then all of a sudden, all production stops because now you can't get the data that you need. Um, so I think for me, like finding that nice balance of like the database helps you when it's there but isn't required to be there to get your file and just keep working. Um, that's something we keep trying for.

B: Right.

I2: [sighs] some–some of the later things that are going on are just like how to deal with uh, more, um, multisite kinds of problems. So, instead of everything being in-house, now you've got Studio X and Studio X Remote Location, and they might need to share data. Or, Studio Beta and Studio Beta Remote Location, or we have a Global Location as well. Um, you know, Studio Delta doesn't really have any vendors for the CG stuff, but they did on Adventure Film Six for 2D. So dealing with data interchange, security, um, being more platform agnostic because sometimes venders are Windows and we're on Linux...so it's nice if you have like–like a web app that like and then you don't care what it's implemented in–you can just go visit the site and do whatever you need to. Um, that's–that's real–a later trend, I guess.

B: Yeah, uh-huh. That's cool. Great. Um, just, you know, was curious about what–what things you'd seen. If there were cropping up patterns like you'd mentioned of things people were trying to implement.

I2: [sighs] yeah, overall, I guess the audience is changing.

B: Yeah.

I2: Like, in the early days, your artists were fairly technical because they had to be. And now it's more and more people that really don't understand, you know, a computing environment–they just kinda know their tools. Uh, and there's only like so far drag-and-drop can take you. Especially when you're dealing with lots of files and lots of data. Um, right. Unless somebody writes you a drag and drop application that you can drop a whole bit–a whole lot of things in, and then rename all your files, it's still going to be faster to just go to the command line and, you know, do something. But that is one of the things people found challenging is you know–like supporting an animator who's usually pretty non-technical. Um, and getting them to publish something correctly. It's like well, they could probably do it more easily if it was drag-and-drop really basic. But uh, you know, and it takes time to write GUIs and all this other stuff. It's like depending on how much time you've got to build the software, it's like well, can you just type publish? We'll take care of everything else. Um, so in the adventure series two pipeline, that's basically what we did. And it worked out pretty well.

B: Yeah.

I2: I think, uh, people take a little time to get used to it, but it was actually less steps than their–or, um, GUI kind of publish method that they had before. So, um, yeah. But then at Studio Beta I was–I can't get people to type for the life of me, so [laughs] more kudos to the Studio Delta animators being flexible. [laughs]

B: Yeah, I don't know. I–I guess I would have just pictured it being the other

way around, uh, with the Studio Delta folks needing more GUI stuff, but that's... interesting. Yeah, and I've–I've–I'm familiar with the Prep pipeline and being, Interview 4 and I built something that was sort of based on Lean, uh, when we worked together last summer, uh, so it's coming in handy now because our current, uh, intern environment for our group project is very similar, so I already have some familiarity with that and that can get all of our artists up to speed and say okay, you have to run this command before you do anything else. Um, you know, so.

I2: [laughs] that's cool. It's nice to see it living on.

B: Yeah. No, it's–It's nice. Um, there are some things about it that I really like–I think it was, um, it might have been a little bit much for our five person project last summer. Um, and of course we wrote like a really small version of it–like a really–the lean, lean pipeline. But uh, even so having like a department based work flow was a little bit of a challenge for just five people because–we didn't–It would have been better to just maybe have like an asset and release area and not have it be so department–linear department based.

I2: Right. Where it's like here's the model directory, and here's the look directory.

B: Yeah. And you have to publish to look and you have to publish to rig and you have to do all that sort of stuff, so, um, I think I–I learned a little bit about scalability. Like what works well with larger environments verses...

I2: Yeah, yeah. I guess that pipeline is sort of a medium sized pipeline–It can scale down, but it's got more overhead than you'd want. And it can scale up to a certain point, but then it's like okay, some of those paradigms start to fall down when you get like a whole feature team looking at it. So...um, yeah.

B: Cool. Okay, well I guess I've only got one more questions then. Um, if you–If you could have a–If you have a wish feature–a feature wish list for a pipeline, or uh,

I guess like a pipeline pipe dream. What goes–what goes in that?

I2: Huh. [pause] well, gosh. Uh, maybe time for more diagrams.

B: [laughs] great. This is all–It's gonna have its own page in the thesis documentation.

I2: We'll see. I dunno, uh, how I'll draw this. I guess I'm starting to think more along the lines of distributed kinds of pipelines and also scalability, but that's–[sighs]. Uh, I don't even know where to start. Okay, so in the middle, we've got some sort of event manager.

B: Okay.

I2: So this helps us–I mean actually we just set up one at Studio Delta. I mean I don't know how widely it's used now, but like if you had, uh, the database–the production tracking database, that's keeping track of status, and then you've got your asset management system, here, uh, a lot of systems–basically, these two talk to each other in order to say hey this status and these things belong and what we'd like to do is decouple that so that I check in something into my asset management system or I change the status in my database, and we can just send and event to the event manager and then it just broadcasts that event out to whatever happens to be listening, and then you can go off and do whatever you need to do. So rather than say, um, I changed the status in the database and I want to trigger like a movie thing... a lot of times you'd have this database have some trigger and it would call a script and it would go make a movie. I would much rather be able to say hey database broadcaster, there's been a status update and I have some tool out there that's listening for that and then it goes off and makes the movie. Like because then, if I also need to make an editorial update, then I can just hang that off of there, too, and have it do its work. Um, rather than coupling myself down to the database.

B: Mmhmm.

I2: We talked about being able to swap out stuff.

B: Right.

I2: Well, what if I need to use Shotgun tomorrow instead of my custom one? Well, then, I would–It's much easier to add things to Shotgun that said broadcast this message than it is to integrate all this other workflow stuff. Um, the other thing that I think I'd like to do is, so–underneath–asset management being part of this stack, so at the very bottom instead of having NFS or sums or a Unix file system, actually build a virtual file system buffer so that we don't care what storage mechanism is underneath here. Uh, Studio Delta already does something kinda like that because, you know, we have fast storage for textures that are gonna get handled by the farm verses slower storage for um, you know, temp files or something like that. Um, but even then it's like we had to–you kind of know like–you have to manage your mounts correctly, um, or symlink into the right thing to say this volume actually points there. Um, and also, when we talked about episodic verses uh, feature directory structures–just differences in naming conventions or whatever, you know, it would be nice not to have to care what my path is, and just have that say, well, um, one of the iss–one of the issues we have right now is when you make a–a directory, depending on where you are, you have to create a virtual volume for the–for the um, net app.

B: Right.

I2: Um, ideally, you wouldn't have to act–ask anyone to do that for you first–you'd just say make this directory and it triggers whatever it needs to do to create the right stuff. So the VFS–virtual file system could help insulate you from all that and talk to a more complicated back end that it's like oh, this data is from Remote Location–and I will download that for you [laughs]. And you know, your directory can show some stubs in the meantime and then–uh, eventually the data gets filled in. Or, you know–basically a lot of that [sighs] if we can put it in the file system, then it

doesn't have to be in the application, right? And so instead of open up this special file dialog in Maya to do the right thing, let's just use the file dial. And under the hood, all the right stuff happens. So that, for me, is uh, also something that could be pretty interesting. So it's–

B: More of that heartbeat stuff.

I2: Yeah, it's really driving stuff lower level and um, shielding people from the complexity of whats's going on so that you have a clean mental model, um, as a user of the pipeline.

B: Awesome. [laughs] that would be pretty cool. Okay, um, well that's all I can think of, um, I guess–you touched a little bit on the differences between like say, episodic production verses–verses feature. I mean, they–obviously they require different things, right–as far as turn around and everything else goes. Can you–can you speak about some of those differences real quick?

I2: Huh. Well, let's see. It's interesting, I mean, it's–because Adventure Series Two is technically a TV pipeline. Um, a lot of it just comes down to the size of the crew and how much time you have to do something. The tasks are mostly the same. Like even for Action Series. The tasks are pretty much the same. It's like you have to do um, story, pre-viz, layout, build the models, rig everything...you just might have two days to turn around something instead of a month. [sighs] Uh, for Action Series, there's an added wrinkle of–you have um, a lot of production happening in Global Location–and at multiple vendors, uh, so you have to basically account for not only your own revision control and asset management, but other studio's, too, and come up with really good standards. I think, especially with outsourcing it's like the more you can come up with a tight specification, the better off you will be–even if people think it's overly restrictive. Um, for the pace that you have to go at, you kind of need to be on rails. You have to be like um, this is it and um, there's these special cases

that you just have to handle the special cases, but the 80% of these stuff is just set then forget. There's, you know–that's the way for better or for worse. Um, because otherwise, you have to have a lot of smart people to deal with all the exceptions, um, and that costs money and time.

B: Right, right.

I2: Um, yeah. I think for TV, more than anything it really is just how much polish can you put into something, um, [sighs] yeah. Feature it's scale–It is the scale and iterations. Right? It's just being able to come back to this like, we thought that sequence was in the can, but they suggest something and we have to go back and revisit. Um, so...yeah. Scalability and just robustness. And to a certain extent, being on rails, again, it uh, just helps with that kind of thing. And so, you want to be flexible in-between shows, um, so you can configure the machine however you need it to be for your particular project. But once you're running, that's like it's kinda locked in. [laughs]

B: It needs to be locked, yeah. That makes sense. Okay. Now–now I'm done. Now I can't think of any more questions.

I2: Well, games is the other–

B: Oh yeah yeah yeah, games.

I2: Um, games is a little bit different because it actually is more like that model that I showed you this–because this is your game engine. So everything has to go into the game engine before it's usable. So you can be working in Maya on your rigs and geometry and everything, but eventually, you have to get it into the game engine. So, a lot more of the pipeline is oriented around a software engineering kind of process where, you happen to be working on an asset–It might be digital asset verses code–but eventually, it'll go into revision control, Perforce or something and then stuffed into the game engine, you know, built and tested um, overnight or something. So, very

different kind of philosophy. But if you look at the artist workflows, they're almost the same as how they would be operating on a feature or a TV show. It's like I still build my geometry and I just have a different publish process and a different way to look at my–my result in the end. Um, so...yeah, to the artist, it's not as much of a change as it is to the data management. It's interesting–because those artists are very used to revision control, [laughs] versus the feature people who are not.

B: Awesome. Okay, well thanks very much for your time.

I2: Yeah.

B: And uh, I appreciate it. This has been very helpful–very useful information despite the background noise we had.

I2: Yeah, hopefully you–If you have any questions, you can just email me.

B: Thank you. I will continue on that as the research goes, so...that is all we need for now.

INTERVIEW 3

BRANDON: Hey I'm here with Interview 3. We're gonna talk a little bit about his work and the, some of the pipeline stuff here at the studio. So Interview 3, if you'd start by talking a little bit about your experience, um, how you got here, that, that kind of stuff. Just a little background so that I can understand where you're coming from.

INTERVIEW 3: Okay. Uh, let's see, I came out of College One, uh, with a computer science degree, and then moved down to Location One for about five years where I worked at VFX Studio One, uh, though out Big VFX Film One and their sort of expansive phase where they moved from being a single-project studio for, uh, John Director's personal films to a more, uh, spread out VFX studio doing stuff for lots of different people. And I stayed with them on through their Big VFX Film Two ramp-up, which was a major revamp of an entire pipeline to be able to deal with, sort of Big VFX Film Two's insane, um, uh, poly-dense systems and all the sort of stuff. Uh, and after, after we finished that pipeline up I left Location One and came back out to Location Two and, uh, was just looking for work and uh, found a good place here at TV Studio One where, uh, had never had any kind of programming, uh, positions at all, never had anyone here who was really focused on putting together the pipeline. It all just grew very very organically around the needs of the people because of this sense of, 'hey, shit, no one's gonna let us keep doing this very much longer.' and it wasn't, you know, itthey were eleven seasons in, and after eleven years I'm like 'whoa, maybe we should start treating this like it's a serious studio and hire some programmers.' and I, uh, I came in, sort of demonstrated that value, and after a year - or, a little bit less than a year, more like eight months - um, we hired two more people, which I sort of became the team lead there. Uh, and then after about

two years of that they gave me, uh, sort of, control over the whole technical means of the studio, and now I operate as the CTO.

B: Wow.

I3: Uh, yeah, keep the whole studio running and keeping the I.T. and programming and engineering, uh, departments sort of connected, moving forward.

B: Very cool. Awesome, well, um, what would you define a pipeline as? I know that's a very broad question.

I3: It's a very broad question. Uh, I would, sort of, because I'm a very, sort of, individual-orientated kind of programmer and that I sort of see, not the problem in an abstract sense and not in a, um, sort of problem-to-be-solved sense but rather as a relationship between two people. Or, in the, in the larger studios more than two people, but between, sort of two departments which you can sort of think of as two people. And then what I define a pipeline al-, as is really just a communication technique between two different, disparate groups of people who have different goals and different needs, and you sort of mediate these need, those needs together based on, usually, um, very specific individual requirements. Uh, maybe not even to, necessarily to, sort of, this, the role, the job, the role of the individual person. And that, uh, it's always very important to me that a pipeline, uh, serves the needs of the individuals who are working on it. And that it's very driven by the artist's needs and the artist's, uh, requests. And so, I, uh, am very very strict about not imposing things from on top and not, sort of saying 'you're gonna work like this' and 'you're gonna do this' and 'you're gonna talk like this' and sort of, a pipeline is to me, I go somebody, an artist, and say 'what do you need?' 'what's annoying?' 'how do you, sort of, what's the easiest way to get this done?' and then I go to the next stage and say, 'well, what do you need?' and you can mediate those two things together.

B: That's interesting. So it's more of a bottom-up approach, as opposed to

I3: Yeah.

B: Right. Because some of the other folks I've talked to at studios, you know, there's kind of like a tech group or whatever, and they define some conventions or something like that and then they kind of pass it down from on high.

I3: Yeah. And some of that stuff you need to pass down from on high, especially things like naming conventions are, it's a good example of something that, look, nobody really cares what it is except for the tech nerds and all you need to do is say 'okay, just do it like this.'

B: Right.

I3: And you just, but you also need to be able to hand that to them in a way that doesn't feel authoritarian, it doesn't make people feel like, 'why do I have to do all this crap?' uh, and, um, just based on the needs of a place like this, where, uh, something that even if it can save you ten minutes down the line, if it takes two minutes, it's a hard sell. Because two additional minutes is two additional minutes that can really matter! [laughs]

B: Yeah.

I3: And, uh, and so we always have to be very very light with our touch and very very, um, uh, responsive to the needs of, on an individual level. But that's something that, it's a luxury we have because we don't have a lot of turnover, as well. I don't have to train someone new to use the pipeline every three days, or every two months. You know, we get a new person in maybe once a year.

B: Wow.

I3: And everybody else is, sort of, they know their job and they know how they're gonna work, so I can work with them in a very, uh, personal way and say, 'man, let's just give this thing whatever it needs to, to make your life easier' while of course always maintaining that sense of 'well all right, let's not get crazy, though,

you can't, I can't give you a one-button solution and I can't let your needs trounce everyone else's farther down the line.' So, yeah.

B: Sure, sure. Awesome. Um, what do you think makes a good pipeline? Because there's lots of different examples of, you know, workflows or directory structures and things like that. What, what makes a pipeline good and useful?

I3: And again, I would go - and I think this one of the ways I'm different from a lot of people who do this kind of thing - I would say the best pipeline is the one that annoys the artist the least. Because, in the end you're not creating a pipeline to serve itself. And a lot of people I've encountered, especially talking, going to industry meetings and talking to people, they get the sort of idea of the pipeline as this living creature that exists to serve its own needs. And I think that's a really really bad way to think about these things because what you end up getting is these people, these artists who just hate the pipeline and they get so frustrated with these little things they've gotta do, and it's got, if it does, if it's a capital p instead of a lowercase p, everything breaks and everything dies. And those little annoyances can create massive frustrations in people's workflows that, uh, I ha-, I love to avoid those things, and I love to go to an artist and say 'you know, you've been working for two weeks, do-, working real hard. What was the thing that annoyed you the most? What were you so frustrated with that you just wanted to shake the computer?' and, and attack those problems directly.

B: Right.

I3: Because I really do think the pipeline that, and[laughs]. This is gonna be a weird phrase, but a pipeline that pipelines least is the best. The one that requires the fewest number of, little, like, annoying changes or, uh, requirements on the artists and the least time, uh, from them outside of their work. Because what they want to do is be as creative as they can, as much as they can. And all that time they have

to spend making the creative stuff, sort of, fit into this mold that you've created, uh, is stuff that, that they hate doing. And, when they hate doing it, they do the things they like less, and they get more frustrated and then the work suffers.

B: And stuff breaks.

I3: Yeah. Yeah. And so, um, the lightest touch, I think is the most important thing in a good pipeline. Uh, and it, I, ideally - and this is almost never possible - but ideally, you can create a really simple workflow where people are conforming to the pipeline and they don't even know they're doing it.

B: Mhm.

I3: All they're doing is just, 'I just know I have to save it in this folder and give it this really simple name' and all this other stuff is done for them, and that, they, they're brought into the system, they're brought into the databases, they've got all their naming, they've got all this sort of stuff that just sort of came naturally as a, um, as a result of them doing their jobs. And one of my favorite things we do here is, uh, as you construct a scene within, um, within Maya, as an example, uh, you just give things the names that work to describe things that they are, and then once you move out of your stage all that stuff gets a m-, automatically ingested into a searchable database that you can sort of find them later and know where they are. And people don't even see that. They don't have to do any work and it's just handed to them. And, um, again that's a luxury of a small studio, more time, uh, but it's great for us, so.

B: Yeah, one of the guys I talked to at a, another studio compared a good pipeline to, like a beating heart. Like, it's there, it's working, but you don't really think about it?

I3: Yeah. That's a good one.

B: Or like breathing. You know, it's something that you're doing, but it comes

naturally to you and it's not really

I3: I would completely agree, I would love to meet that guy and have a drink with him. [laughs] because I've met a lot of people who don't think like that, and, uh, but that, yeah, that's what I think is the best one. That it's, that it comes naturally.

B: What kind of, uh, feedback loops, or communication tools are used in a pipeline, or are critical for a pipeline, do you think?

I3: I think in a lot of ways that can depend on the size of the pipeline and, umthe critical ones for us is physical communication. [laughs] And that, that's another one of those things that people can forget, and sort of try and engineer away the need to ever talk to anyone. And, but, um, I've found, those are the loops, and those are the feedback loops that hit the most, um, the most sort of points of communication. Because you know, of course you always get more that way. The sarcasm comes through, the, the, all that sort of stuff. And that, to, to not eliminate those is crucial. And when you have people working in different buildings or stuff like that it's important to have some, some point of physical communication.

B: Ok.

I3: But, that being said. Uh, the tools that we use, that are most important, umand again, uh, I know we want to be agnostic, but the feedback loops and sort of the um, the return to, uh, of information to the artists for instance, and uh, what we call retakes and where they go through the exact same, um, uh, iterations of the shot at the most times, umit's important that, again, be reasonably transparent to the people doing the reviewing, as far as feedback loops, and, um, uh, retakes. And that, sort of, you can w-, because the way our pipeline is designed, people will just do their work, it automatically appears in the list of to-be-checked, it automatically appears in the list of here-are-all-the-notes-for-it, and then it just goes back out to them, so it's all very, um, very automated, but most of that, that's easy. There's nothing very

revolutionary there.

B: Right.

I3: What we find, are, and one thing we're working on now that is, sort of, more interesting is that animation notes don't come through very well, um, in text. And, the, I know lots of animators who constantly are complaining about how they get these notes that come back and they, they think they address the notes and then a the note comes back and it's either the same thing or it's like, 'more excited!' and you sort of, like, 'well you just said more excited before, and now you're doing it with capital letters!' and sort of, what does that really mean? Um, we've got now an interesting setup where we actually hooked a Kinect up to one of our rigs.

B: Oh, wow.

I3: Where - and we're gonna try implementing this, this is brand new - we're gonna try implementing it now where they sort of, where Sam Director or one of the other animation directors can act out what they're looking for, and sort of go, like, 'It wants to be like this!' and sort of, and it'll actually give you, not, of course not final animation but this sort of sense of what it is. Because, again going back to that human interaction, the most valuable notes we get don't come from anything that's ever written down. It comes from the animation director going over to the animator and literally acting out what he wants. Because he's in there with the creators, he's seeing what he wants and he just goes, 'okay, now it wants to be like this. When he's digging it goes in and it stays, but then he sort of shoves it sideways because he's being a really crappy digger.' and all this sort of, those kinds of notes, um, don't come through very well in text, and we're, we're working on ways, again with the Kinect and all that, to, to digitize that but also make it recorded and not have it just be in someone's head. But it's, uh, it's hard to do because, you know, feedback is the most important thing, especially at the animation level.

B: Right.

I3: Uh, but there's just, uh, not much, not much that can really convey that really important information.

B: Mhm. So, email is kind of, like, on the back burner then?

I3: We don't email at all. We use IM, chat a little bit, but not much. Most of it is just, um, it comes through our little proprietary system which is essentially the same as email, but, uh, the text-based notes, unless they're really simple, like, 'change that jacket to be green,' um, they don't capture, the notion of it. And, um, and you can't, and they never will, I don't think. I don't think you can get that in text. Uh, and sometimes you can work with audio, you can work with other kinds of systems that try to capture more of the cues, but really they're notI don't, I don't think there's a solution there, really. I'm, I, uh, I would love for there to be one. But, it's a, a, yeah. It's an ongoing problem, I think.

B: And, um, to - obviously something, you know, any kind of digital media, any kind of show is gonna have lots of iterations, uh, for shots, for sequences, things like that. What - and, and the assets as well probably go through different phases - uh, how do you keep up with version tracking? And how do you, um..I guess the question is, what are some good methods for version tracking? Like, ways to keep up with those things.

I3: Um, we, we work with a very specific, what we call, what I call the moving forward system here, which is essentially a heroing system where you have the heroes, and then subversions that sort of, uh that are just backtracked a little, but we almost never go backwards in versions. It's always going forward. So you're never gonna have anything where it's like, 'well version 3 is the hero, but there are 6 other versions that, you know, we're not working with' and you sort of work with symlinks and all these sort of complicated systems to make the artist not, to, to make that transparent and

to make the artist not have to see that. Um, because we iterate so quickly - and we iterate through a shot in, um, usually we, yeah we can go from script to ready shot, like, airable, in under an hour.

B: Wow.

I3: Which is, yeah. That's the, that's it's literally been storyboarded, created, had the artists you know, the, had the, um, concept art created, had that brought into Maya, rigged, animated, done, out there, in an hour.

B: Wow.

I3: It can be done. And, the, so that means a lot for most of our stuff. There's not any real concern with, 'well what's the right version?' because it was all there, and it was all, we all worked on it in, uh, you know, in a very shot amount of time. You don't have those weeks and weeks and weeks of, 'well it got changed, which note is the right note? What's the newest one?' so what we do with that simplified system, and again, because we never want to create complications. We never want to create a needless complexity, that we don't really bother, almost, with versioning. [laughs]

B: Yeah. [laughs]

I3: Uh, we just sort of say, 'look, if there's a change. It's just version6.' you just go forward. And if there's, you know, and if we end up with 30 versions of a thing even though we've only been working on it for four days, that's, you know, that's fine, um, because it's just, you know, it's version five, it's, it's in the naming convention, it's in the, uh, system, but really it's just, the thing is the thing and you go forward with using it. Um, and as much as anything we just - even if more notes come down there comes a point when the producers tell the creators or anyone, they say, 'look, no more versions of this. You can't have more, we don't have time, there's just, no time to work with that.' um, so, I think that is another unique thing about this, is that we don't have time to work with that, we don't reallyI know it's a huge issue

for so many other places, uhbut for us, it's so, it's so minuscule a part of our system because, um, it just goes forward so quickly. Yeah.

B: Yeah. Well I can imagine, you know, like, at Studio Delta, they have, something like 180 characters in Big Animated Film, you know, and it's just such a massive thing to keep up with, but, um, I guess the advantage of being smaller is that, you, that you don't necessarily have to keep up with that number of assets or anything like that.

I3: Yeah. And it's also, I mean it's worth mentioning that our, our heavy asset, den-, our asset-dense shows, which happen occasionally, we have lots and lots of new stuff, there's never more than one version. Because it doesn't, like it, if you wanted a change to it - unless it was a really crucial, main character sort of thing, there's absolutely no time for us to make a second version, because there's ten other things that have got to, that have to be worked on, have to be converted to Maya, have to be rigged, have to be done with, have to have all this stuff done to them so that, um, uh, the things that you would think of as being the most difficult are in fact the easiest, because whatever the ver-, whatever it is, it is.

B: Yeah.

I3: Never time for a second version. Um, whereas the light things, like, where we maybe only have two or three main characters, or just the boys, that can be a concern because he'll want different things, we want to make sure we get the right ones, but it's not that big a deal just because we have our, you know, our basic library and whatever the shot is, the shot is. [laughs]

B: And I think there's probably, uh, something about the aesthetic of the show that you can get away with that pretty easily too.

I3: Some of it, yeah. I mean, we definitely make mistakes sometimes. Uh, and, get called out on it by the fansbut, uh, it's not that big a deal.

B: Yeah. Cool. Um, well, I haven't seen 'Behind-the-Scenes Movie,' but uh, I thought about watching it before I came over, because I didn't know if there was anything relevant in it or not. But

I3: Yeah, I guess not, it's, it's very much a, uh, it's about the creative process, not the technical process of making it. Um, it's a - which is a unique process, just given that, even, even Behind-the-Scenes Movie is a little bit of a misnomer because generally those first two days of work get thrown out. So we do it in about four days of really intense work. I mean, the last three days are essentially, you know, 18-hour day, sleep, come right back in. 20-hour day, sleep, come right back in, 30-hour day. [laughs]

B: Wow.

I3: And then sleep for two days, come back in and do it again.

B: [laughs]

Umand, and, that level of, of, just, intense crunch time means two important things for us programming. One is that we don't have test time.

B: Hm. Yeah.

I3: And I mean that. There's no time to test any of our code. Uh, we get, you know, four months off in the summer, when nobody's here it's like this. We're, it's just us, the programmers, I.T. crew, we're working our asses off doing everything we can. But, we're not really using the systems. No one's testing anything. There's no time to do, like, a versioning of code, or like, you have, you put it out there, production-ready, stuff that hasn't been touched in three months and then you say 'okay, what's wrong? Okay, fix it.' development cycles happen in hours, and you sort of say, 'okay I want a whole new version of this that does this, this, and this, you have four hours. Do it, get it.'

B: Wow.

I3: And then, uh, and it's gotta be working, and you've gotta be able to make sure people can do their job and, and make sure everything goes forward in a time that is just ludicrous, because there's just no opportunity and any other way. And then, of course, once the run starts, everything is locked down. Unless it's a major bug, we don't touch the code.

B: Right.

I3: Uh, we're just here making sure everything goes forward and works. And, uh, that's unfair to the developers to a point that, no one really understands how unfair that is here, except the developers. We're sort of like, okay, this is exact, if you were to write a course on how not to develop software, you'd do it like we do it here.

B: Right.

I3: Uh, and yet that's how we do it. And, uh, and everything we do has to have a, this, really, this thing I keep telling our developers, that, uh, you know, if it, even if it gains you ten minutes, it can't cost someone else two minutes. Every second counts, and every second that we add to somebody's workflow, unless it is game-changingly good, it can't be added. It's gotta take away, it's gotta take away. Uh, and which we've managed to do. And at this point we've got, you know, our iterations are insane, but we've managed, since in the last three years which we've completely re-written the pipeline, we've taken it down a whole day of time. So, it's been really great.

B: And I guess, with that level of, of rapid development you probably need a lot of flexibility in the pipeline.

I3: Absolutely. All of our in-house tools are, are in-house written. We don't make use of any external code, uh, well, I - that's not, to be fair, we use external source code, bring it in and work with it, but we don't use any products in our pipeline that are not custom-built. We tried for a long time, or before I got here they

did, and, um, and it was just a nightmare. It, just cause, no one could appreciate the fact that when you, once you find a bug that adds two minutes to your workflow, that'sthat's a game-changer.

B: Right.

I3: And you can't, you can't call someone at a development house, like, who's running Shotgun, who's putting together Katana or something like that, which are awesome systems, but if you find stuff and it breaks, and you're like, 'I need this fixed today. Absolutely, I need a whole new revision, a whole new build, I need it fixed.' they'll, they laugh at you. They, they're right to laugh at you, but, umour biggest, uh, piece of a-, non-in-house software is Maya, and, uh, it's a scary, scary beast. Because even as customizable and programmable as Maya is, it's not enough.

B: Mhm.

I3: It's not enough to really have core access to that machine and be able to do whatever you can with it. Butyeah. That's, sort of where we are.

B: Awesome. Um, well we talked a little bit about, my next question was, sort of, thinking about, you know, pipeline decisions and stuff thatwe kinda talked a little bit about that earlier, like there's, there's not necessarily a, uh, a group of, a secret cabal of people that, like, hands down everything.

I3: Yeah.

B: We, we talked about that a little bit. Um, can you talk a little bit about the evolution of the pipeline, uh since you've gotten here, things that, things that, um, have changed for the better, and things that maybe you miss or that, um, have, like, has it gotten better? Has it

I3: Well it's ju-, I mean, it wasn't anything when I got here. It was completely cobbled-together from code written by someone who had taught themselves to code, who like, who wasn't a programmer. He was doing the best he could with what he

had.

B: Right.

I3: But, um ,and I, I, like out-of-house software and sort of, whatever they could sort of find that would allow them to do the job. Now we have a much more streamlined system, so it's, it's much much better. But there's always room for improvement of course.

B: Mhm.

I3: But, um, the evolution was really, um, it wasn't as natural as I could have hoped because, you know we, there was no ramp-up to try out the system. It had to be very, um, very component-based. So, you know, we sort of started with assets, and we moved on to shot tracking, and now we're sort of, and we moved on to revision tracking, and sort of, all, uh, internal asset tracking within, assets within shots. And now we're looking at management from a more, um, generalized perspective. Um, but really what it all came back, what it all comes back to, uh - and I think this is always the case - is the database. The database is the heart of any pipeline.

B: Yeah.

I3: And, um, we keep going through revisions of the database to sort of smooth it out and make it simpler and faster just because we don't have a very complex, sort of and the need for a complex system. Uh, but, the um, the database sort of, it became the heart, we bring up some, sort of, easier protocols around that, and then we focus on, again, the individual users and what they need, and what we need to do the shows. And, and really what we looked at initially was, 'what's taking the most time? What's the most annoying thing, the thing that breaks the most, the thing that sort of causes us to, you know, rack our, rack our brains?' along the entire pipeline, let's slot something in that works for them, focus on getting them working. Ok, what's the next thing? We're gonna slot something in, we're gonna

make sure it talks to this other thing. Ok, let's, what's the next, what's the next. And that's what we ended up doing, is on an application level, we slotted those in individually as we moved forward, just making sure each application was able to sort of communicate with all the other stages of, of the pipeline, until, eventually, and it - it was interesting, because, to a lot of people at first, if sort of seemed like, 'I just have one little thing here, he's got one little thing here, there's nothing really going on.' it wasn't until the whole system, like, you know you've got the last stone in the arch into place that suddenly it was like 'oh, ok now it's all connected, it's all talking to each other, and we have this streamlined thing.'

B: Nice.

I3: But yeah, for the first two years it was very much, let's just solve individual small problems that are causing issues and then sort of use that to expand the eventual idea of what we want to have. But that's uh, what I would say is the most important thing for evolving a pipeline, is to not be obsessed with the entire thing, but rather to focus on the individual small, solvable problems that you can work on. Then, but with the view, with you, being the pipeline designer, having the eye towards the eventual completion but really, you know, solve the individual problems first.

B: Right. Not coming in and saying, 'we have to throw all of this out and start over!'

I3: [laughs] yeah, exactly. That will cause more headaches than it's worth. Yeah.

B: Sure. Yeah. Uh, I guess I just have a couple more questions. I'll throw some, some ideas at you that some of the other interviewees mentioned. Um, one of them was smart assets, and using metadata in assets.

I3: Mhm.

B: Um, what do you, like he, he was talking about that, sort of becoming a bigger part of, of the pipeline at Studio Alpha, and, um, trying to, um, you know,

just add more and more metadata to characters so that you can pull things, specific assets like, say, from, the version you used from a trailer or something like that.

I3: Yeah. And that sort of stuff is really, is really great and I'm, um, sort ofwe ended up sort of writing a system like that back when I was at VFX Studio One and then that was a really important aspect of what I wrote, was sort of, model packages. Models come not as an individual, um, like, Maya file as you might think of, but rather as this big old grouping of metadata, and different versions and all the, like different polygon density versions so you can work with them and all these sort of things, that, now is pretty standard. A lot of, really everyone uses it. Back, you know, 5 or 6 years ago, it was very new and no one was really working that way.

B: Yeah.

I3: Um, again, the, uh, the problem I have with those sort of model packages - it, it's not a problem, it's just a statement of what I think is a statement of fact, you can take it or leave it - um, is that, they require a great deal of human intervention to manage, and that it's not. That, that idea I had of, or of, that I believe strongly in of the transparency of the, of the pipeline, um, means that, you know, you don't want to have people having to sort of manage their own assets too strongly, and like, ok it's this, and it came from this and it has this in it. Um, it's hard to, sort of, to keep the information from and automated system, uh, relevant and, uh, searchable, which is a, you know, the most important thing is that you gotta be able to find it later. And being able to sort of say, being able to find that trailer asset, like you said, um, is, you know, that's as important or more important than having it actually be recorded, because the biggest problem we had with asset packages and tracking assets was not getting them into the system. That's easy.

B: Mhm.

I3: It was making that system usable in such a way that people could easily find

what they were looking for without having to be like, 'oh, you know, where do we find that beaten-up version of Main Guy? Where is that?' 'oh, well, he was in this show, well and we beat him up at that time.' and that sort of thinking is not helpful. But also, like, you know, the reason we couldn't find him is because it wasn't called beaten up, it was called, you know, uh, destroyed or something like that, you know.

B: Yeah.

I3: And, uh, working with things like synonyms, and stuff like that can be a major problem. We're moving right now into one of our new projects, into what we call the Show Location map. And that's where we're making an asset system that's subordinate to geography, and sort of saying that, 'we've got this really, we, this unique system where we have a very specific area, we're working with Show Location. Let's, instead of working with trying to find search-ability via text, which is sort of the accepted method, we're gonna move it to a map where we literally have a Google Map of Show Location and we say, 'If you want to find something, it would be wherever it is on the map.' So if you want to find the school, you go to the school on the map. If you want to find businesses, go to the business district, and all that sort of thing. And it provides a kind of easy shared reality that um, allows assets to be much more easily found and categorized, because you don't have to think about the ten things that you might be thinking of later, all you've gotta think of is where it would be, that's the only thing that's important. And that provides a, what I hope will be a very, um, unique opportunity to be able to find things and search for them in an easy way.

B: Yeah, that's really cool. I've never heard of any, anybody doing anything like that.

I3: I don't think we've, uh, like, most places wouldn't be able to.

B: Well, sure.

I3: Because like, yeah, there aren't many, ten, you know, animated shows with ten years of backlogged assets that, so, uhbut I think that's, that will be a good thing that we're gonna move forward on, yeah. So, yeah, we're testing it out this run, as well. So we'll see how it works.

B: Very cool. Um, I guess the only other thing, uh, I would ask about is, umthe second guy I interviewed talked about, um, he drew a little diagram talking about checking, checking things in and out, basically, like, having, um, like an asset repository almost, where people come in and they check something out, they do something with it and then put it back in, and then the other piece, like, can grab that and, and start working on it.

I3: Yeah. And that's, umI think those are a mistake, personally. I think they're a mistake, not because they're not good ideas, they are good ideas. They're how all software development is done now, but you've got to have a software developer's perspective on how that works to do that. Uh, and this is one of those things that, especially programmers forget a lot of the time, which was how conceptually difficult were some of the things that they now think are just completely obvious were to grasp initially. Because if you ask a programmer 'how long did it take you to grasp pointers the very first time you were told about them?' it took some people months, of, of like really trying to figure out, 'what does this even mean? What does it?' and, um, versioning systems, to me, work the same way in that the first time I was trying to work with a versioning system and sort of conceptualize what 'checking out' even meant, and how that sort of, and how merging even worked, and all that, which is now, it, it's obvious to me. But back then, it was very very complicated. And getting, um, artists conceptually, to sort of, to view a) the potential of that system and b) how to not screw it up, can be really really hard. And I, uh, I have never seen the benefit of, of forcing that system on them because it frustrates the crap out of them.

B: Mhm.

I3: And even though it's sub, you know, it's suboptimal not to use it because they're so powerful, there's so much you can do with them and it prevents so much trouble, it allows for these automatic, um, ingests of, of new versions into, sort of, older versions, and all this great stuff. That's great - if everyone knows how it works and it works properly. But in the instances where it doesn't, man, it creates major headaches. And I, uh, that's just why I, I would be very reluctant to implement a system like that. Uh, because, man, even if you know how it works, if the artist doesn't it's no use to anybody. But, uh, but they're good systems, they're really neat, but, uh, they're scary. [laughs]

B: Yeah. Well that made me think of another, another question, umon artist usability and making things, you know, easy for them, do, are the people at TV Studio One required to do any command-line stuff, or is there, is there a lot of, is everything GUI-based?

I3: It's all GUI-based. And it's interesting that you bring that up, because it's one of thesethis is a conversation that I had three years ago at a tech meet up that was talking about this very thing. And the, the banner headline of this whole talk, from this very, like, well-known guy, who was doing some major pipeline stuff at sony, was, 'artists have to be comfortable with command lines.' and I said, 'well all right, that's great - no they're not!' [laughs] you can scream that from the rafters until the day you die, and they will not be. And they'll be frustrated with it and they'll call your support staff, and I, and they will say, you know, you can'twell, let me put it this way. In my environment I can't enforce that. It's just not possible. Uh, I can't, I don't have the support staff to deal with the questions that would come in from 'why doesn't this work? Why isn't this thing doing what I want to do?' I can't, um, I don't have the time to train, to pay the artist to train them in how to do it, because,

you know, we've only got these set amount of time, and we're working and they need to be on the ground rolling. And, um, and at this point anyway, you knowfive years ago, eight years ago, when I started in this business, maybe that's something you needed. That's not true anymore. Writing GUI tools is fast, it's easy, it's not hard. Um, handing all, writing ugly-ass code - using things like regular expressions and all those sort of text-processing things, it used to be a big no-no because they were so slow, they would slow down your tools - is not true anymore.

B: Right.

I3: You can write ugly-ass programs with workable UIs that, um, that anyone can use and will send, will not be an issue. And, and I can't go to my producers and say, 'look, they have to be comfortable with the-, UI, with a command line because I don't have any options' or 'there's no way for me to do my job unless they can use the command line.'

B: Right.

I3: I don't, just, don't think that's true anymore. Um, and, you know, if you can, if you can make them do that, that's fine. It'll save your programmers time, but not much. And, uh, and I think the time you'll lose to support calls, and, again if you've got ten wranglers doing support calls 24 hours a day, then that, that's not an issue for you. For me, at a small studio like this, no way I can handle that level of calls, so I've gotta be able to just sort of say, 'look - it's all GUI. Everything.' it's um, they don't even know where the terminal is, most of my artists. And, uh, and really even though I very rigorously ride the programmers, saying, 'everything is terminal, command line interface to terminal.' so like, we can run all of our tools from the command line and for de, debugging, we can separate all of our tasks from, 'this is a GUI issue slash this is a command line issue.' That's very very important.

B: Yeah, definitely.

I3: So that, for debugging we have that kind of, um, that capability, but, uh, the artists don't even know it's there. And they'll, they never will. [laughs]

B: [laughs]

I3: So yeah, that's where we are on that one. But I, uh, I don't want to sound like a militant about that, I would love to be able to use the command line for some things, just because it's faster, it's easier, and once they're used to it, it'll be, it'll make the artist's life use, easier. But, not here. [laughs]

B: Yeah. Okay, well, that's all I can think of. Um, thank you very much for your time, I appreciate it.

I3: Yeah, thanks for coming by.

INTERVIEW 4

BRANDON: Hey I'm on with Interview 4 at Studio Delta. Um, we're gonna chat a little bit about, uh, some pipeline stuff. So, Interview 4, uh, to start with, I have some questions about, uh, like, your background, um, just so we can get an idea of where you're coming from, uh, how long have you been at Studio Delta?

INTERVIEW 4: Fifteen years this month.

B: Oh wow, congratulations.

I4: [laughs] I started here right after Big VFX Film Three, so, [90s].

B: And you, so you worked on Big VFX Film Three. You were at...

I4: I was at VFX Studio One before that for three years. Three years-ish, yeah.

B: And, um, your education, stuff like that, where...

I4: I did my undergraduate in computer science, astronomy, math, physics at [university 1], and, uh, I did my [graduate] work at [university two], in computer science.

B: So what brought you to VFX Studio One?

I4: Um, I, I actually since high school knew I wanted to make movies, so I, uh, and wanted to get into computer generated imagery, so, um, my undergraduate-graduate I was aiming in that direction. Uh, while I was a student at [university 1] I started volunteering at SIGGRAPH as a student volunteer, uh, so I was a student volunteer for, uh, [the early 90s], uh, and then during those years I was making a lot of contacts, you know, meeting folks and, uh, and sort of, you know, the research work I was doing in grad school got me noticed, and, so I had VFX Studio Two and VFX Studio One both talking to me my final year of grad school, so.

B: Cool.

I4: And then I ended up deciding on VFX Studio One because they were just

barely started. They had, uh, they were working on Action Film when I was out there interviewing with them, and they just, it was such a young company, they were hiring a lot of A-level talent to do A-level work, but they weren't set in their ways and everything. It seemed like a good place to come in and do cool stuff. You know, had a lot of flexibility.

B: Yeah. So why, why jump from there to Studio Delta, after Big VFX Film Three?

I4: Um...there's a bunch of reasons. Visual effects houses tend to be kind of, they, they're service bureaus by nature. Uh, they bid against one another to land the work, they undercut one another a lot of the time so they end up getting very little money, very little time to do the work. Uh, it turns into not a lot of opportunity to do things right. You end up having to do things however you can get them done. And, there's, I mean there's some appeal to that. There's some challenge and some fun to that but there's also a lot of blood on the floor when it's done, so, uh, it culminated, I...Studio Delta had been after me for a while, they started talking to me [early] about coming over, and I had, I had declined the first few conversations and then I rode this wave of production from Blockbuster Film to Action Film to Big VFX Film Three, uh, where by the, by the crest of that wave I was working 100-hour weeks.

B: Wow.

I4: And, uh, for several months, and my daughter went from 12 months old to 16 months old without me seeing her awake. And, I just said, I don't want to do this, and, uh, looking at what Studio Delta had to offer, the notion that every part of the production process is done in-house, you know, from the germ of a story idea all the way to color timing, uh, meant that they had, they had more time, and they had the ability to do it right, or to try to anyways, so. There's more planning involved, there's more structure involved, and that translates into, uh, a little bit less abuse.

You know, uh, a little bit saner work environment.

B: Mhm.

I4: The production environment's crazy to begin with, and there's an aspect of that that's necessary because you're making art. You know, if there isn't some amount of blood on the floor then you weren't working hard enough, is the mentality I think everyone in this business shares to varying degrees. So, but at Studio Delta we work, you know we get into our crunch period, we get up into 60 and 70 hour weeks for, you know, for some number of weeks at the crest of the production, it gets a little bit crunchy. But, um, it's still considerably saner than a lot of our peers in visual effects. Um, and that's part of the appeal. Uh, there's a lot of other aspects, but that's certainly a piece of it. I get to go home and see my kids, so.

B: Yeah. Well that's good. Um, I guess then, you'll have a little perspective from both of those approaches at work and maybe some of the differences in the way that, they, they do things. Um...

I4: Well, also in my role here, I, it's part of my role here to understand how other facilities are functioning. Um, because, we're, we're always trying to improve, I mean, as are, as are, every other facility who do the same work, we're all trying to learn from one another, all trying to get better at our craft, and a lot of the questions don't have right answers necessarily. The answer changes. If there was one right answer as to how to build a pipeline or how to build a toolset, then somebody would have done it by now and we'd all be using the same tools. And we're not, and there's reasons for that, so. Um, we're always reevaluating, we're always seeing how we can do better the things we do, and how can we learn to do new things. So.

B: Well based on your work, and your experience, how would you define a pipeline? And I know that's a really broad, broad thing to throw out there, but...

I4: It means a lot of things to a lot of different people, I mean, for us, the pipeline

is kind of the backbone for the production process. It's the, uh, we have individual departments, we're a relatively large house, so we tend to compartmentalize. Uh, the smaller mom and pop shops, knocking out commercials, things like that don't do that nearly as much. They could have small groups of people that are working completely freeform. But, uh, just like, you know, small-scale auto production goes from a group of guys standing around a frame, you know, bolting parts on to something like the assembly line. Uh, the bigger houses have moved towards, uh, a compartmentalized, structured approach to the tasks and who does what, and when, uh just because it makes them more efficient. And, at the end of the door when we're clocking out, an entire animated movie that has 1600 shots that have a variety of different challenges to them, uh, the more sane and structured we keep the production process, the better. And so, to me, the pipeline is about all of that. It's about, how do we get a shot from, you know, from a bunch of drawings coming out of story and visual development, uh, into assets and shots and all the way out to the 2K frame. Um, and there's, you know, a vast variety of components involved in that, so.

B: Mhm. So there's obviously lots of different approaches and, and things for a pipeline. What do you think makes for a good, or an effective pipeline?

I4: [laughs] um, it's funny, I mean, my definition of that's changed over the years, several times over. We've gone through a number of pipelines here. Uh, there was a period of time where, and, and this is somewhat sad to say, but there was a period of time where each show was its own little island to some extent. There wasn't a lot of cross-show communication, and, uh, a team would roll onto the show relatively early, uh, they didn't know a lot about what story they were making yet, because the story was still way too early and they, and, time and again if you started into asset creation too early, you'd end up getting very bitten, because you'd end up burning a lot of resources building assets that would end up not being in the movie.

B: Right.

I4: Just because the movie changed. Um, so you try to delay that as long as you can. At some point you have to start building things, or you run out of time to get the movie made, so you have to start making guesses as to what you think has the best shot of being in the movie and start building. But there's still this period of time early on, a number of years ago where you had a team of really smart, really creative people who would be rolled on, and they didn't know what movie they were making. And, pipeline becomes and easy target. They start to think about the things that were hard on the previous movie, or two movies ago, and they start to think about, 'well, we could do that differently, we could try this new way of doing this.' and, that's not, that's not a horrible thing in and of itself, but it did translate into a lot of churn. A lot of, you know, reinventing big pieces of the production process with every movie. And we learned a lot from it, but the term we coined at the time, kind of, watching all this, time and again, we kept making pipelines that were differently broken.

B: [laughs]

I4: Uh, as I said earlier, no pipeline is perfect, and every pipeline has its strengths and its weaknesses and so we kept replacing components with components that had different flaws. They were stronger at some of the things the previous one had been weak at, but they were weak at new things. And in some cases, things that clearly the previous pipeline had been better at. Um, and then you would revisit and you would go, 'oh, well let's try to do that and that, and be strong at both of those things,' and then a third thing you'd be weak at. Um, because, you know, again - if there was a right way to do it, we'd all be doing it.

B: Sure.

I4: There's, every approach we take is going to have those strengths and weak-

nesses, and you just try to arrive at something that's good enough. At the end of the day, it needs to be, uh, the pipeline needs to be good enough to get the data through without undue hardship to the artists, and it needs to be strong enough to, for the, uh, we refer to it a lot in, in the last few years, the 80% case. You know, 80% of your work should go through effortlessly, and the last 20%, the more you try to optimize, improve, all that, the - you end up causing, uh, your systems become bloated you end up causing, you know, these pathological cases, you start to chase down, start to introduce weaknesses in other things. And so we ended up in recent years, two main differences here. One was moving towards a single pipeline for all productions where we are evolving the toolset in a concerted manner and more of a continuum of development between productions, so each production is, is constantly communicating with the show behind them and the show after them about the changes they're making and, and, there's some degree of consensus between them. That's one aspect, and the other is that recognition of the 80% case, that there, we're gonna try and build a pipeline that is effective for the 80% case and just be okay having to handle the other 20%. Be okay with the, the, keeping the pipeline relatively lightweight, and have some degree of hand-holding. And, and knowing that because we kept it relatively lightweight, we'll have the bandwidth to do that hand-holding for the things that don't fit.

B: Sure.

I4: So, that's sort of where we've landed, in real broad strokes, with our pipeline. And it, and it's already panned out, you know, the, having a singular pipeline over, over several productions in a row now has led to a strengthening of overall artist effectiveness, having a pipeline that's relatively light on its feet has made us very nimble, uh, with challenges that have popped up that we had no way of anticipating. Uh, and it's enabled us, as more shows are ramping up, to put more energy and

thought toward the new cool stuff because we're not reinventing pipeline and process.

B: And I imagine it's easier for artists too, to be able to go in the same environment between shows.

I4: Absolutely. They can go from one show to the other. When artists rolled off of Funny Movie onto Family Movie, they could immediately start lighting test shots and started to play around with the new challenges of Family Movie, because the tools that were starting to be built for those new challenges were layered on top of a toolset they were familiar with and made it considerably easier because the tools were very very much the same.

B: Mhm. Uh, what kinds of, well, communication's a big part of a pipeline. Um, what methods of communication, what kind of feedback loops do you, do you find effective, uh, at Studio Delta?

I4: Um, as I mentioned earlier we, we like planning, uh, we like, because we have a longer turnaround on our projects. We have, you know, sometimes three years for a movie instead of - a visual effects production, for instance, will sometimes, sometimes land the project when principal photography is already halfway done and they are in crisis mode from day one - we have a lot of lead time to, well not a lot, but some amount of lead time to plan the movie, so we start to lay down our communication pipes relatively early. Um, the production management team in particular, because they're the ones at the end of the day that are tracking the progress of data. Uh, from my perspective we're worried more about the quality of data and the survivability of data, that it get through the pipe intact and it arrive without flaw.

B: Right.

I4: They track the productivity of data, the, the artists in a given department are clocking out work at a sufficient pace to get the movie made on time. And so, they, they do a ton of tracking and communication. Um, we use, we use shotgun as a

production foundation, um, and then have layered a ton of custom tools on top of it to, to, facilitate the different things they want to query, the different kinds of reports they want to build so that they're communicating to everybody exactly the state of the show from the various perspectives people want to see it from.

B: So, um, would something like email be, a big part of how people keep track of things, or...?

I4: Email's absolutely a way for that, um, we've done a variety over the years, you know, web forms and all kinds of...uh, email, we, and we do, today, all of those things. You know, we'll populate web forms with charts, we'll, um, and we do emails, we do pages, you know, we do a lot of different ways of passing information along. Uh, nightly reports and graphs, and all kinds of things you pull up when you get in in the morning. Email is fairly intrusive, um, so those of us who are, kind of, in that inner loop where we're being, we're in a spot we need to be communicated to a lot, uh, we end up being very very good at writing filters, um because I get on the order of probably 600 emails a day.

B: Sure.

I4: Um, and some are really really important and absolutely need my attention immediately, and email is absolutely a good communication mechanism for that. Uh, that's why, you know, a lot of us are carrying around our iPhones and iPads and are in constant communication. But, there's also a lot of it that is stuff that you need to know but you don't need to know right that instant, and a lot of it gets, we filter away into folders and when we get back to our desk, we take a look whenever we have a moment to breathe, uh, we take a look at the other things we need to be paying attention to. Or, for instance, things that we know need to be paid attention to but we also know there are people doing it for us, right? Uh, for instance, the main frontline support mailing list, the prod support mailing list. I filter that into

a folder and I look at it when I can, because I'm busy all day long running around, I'm not on the floor solving those problems, but I need to be aware of them and I need to at least have, you know, the pulse of where we're at within an hour or two. And so whenever I get a free moment, I'll crack open that subfolder where all those got filtered to and I'll read them, at, at, not at my leisure because that...leisure is a luxury we don't have, but when I have a moment to breathe I will crack that open and read it. But, so, anything that survives my filters and makes it into my inbox, uh, becomes something worth immediate action, so.

B: Mhm. Well, the reason I ask about email is cause I, some of the people that I've talked to, you know, like, uh, like Interview 1 at Studio Alpha, and, and Interview 2, were talking about email, um, and had varying opinions on its effectiveness.

I4: Yup. It's really easy to get buried.

B: Yeah, it's noisy. Interview 2, that was Interview 2's main point, is that it is inherently noisy. And, uh...

I4: And so we filter the hell out of it.

B: Right, right. Well I talked to, um, to Interview 3, uh, over at TV Studio One yesterday, and, they, they don't use email at all, which, I found interesting, um, but I guess you can probably get away with that in a studio that's much smaller.

I4: Uh, there cert-, I mean there's certainly other communication mechanisms that are functional, you know, like I said, we have graphs that we look at every morning, we have web forms that are populated constantly so if you're at your desk you can pop them open, we have, uh, iPhone apps to look at the status of the queue, you know, we have a, we have a lot of things, um, other ways of communicating, but email is deliberately intrusive. You know, when you open up your iPhone and you see that box has five unread messages, and you know, like I said, that your filters are throwing away all the stuff that doesn't require your attention, you have to open it.

You have to open that up and look at it right in that moment, and so it's, you know, shy of texting people it's the, it's the most intrusive form of communication, and so, there's times when that's necessary, when you have to get people's attention.

B: Yeah.

I4: Um, worst case are things like, leaving people voicemail. You know, there's, I, there are days I never see my desk, because I'm just, I, I get in in the morning and I'm immediately running off to go do stuff, and I just never make it to my desk, and so there have been days where I make it to my desk the next day and see that red light on my phone that's glaring back at me and go, 'ah, crap.' because yesterday something happened that needed my attention and I didn't even know about it because I never made it to my desk to check it. So, we've definitely gotten more mobile. Uh, a decade ago we were using walkie-talkies...

B: Oh, wow.

I4: Uh, we were carrying walkie-talkies around the building just specifically so you could be grabbed at a moment's notice, so somebody could get your attention to ask a question. Uh, email, email is the primary method for that now and then texting is the crisis mode, 'i need you right this second,' uh, mode of communication for most of us.

B: Yeah, yeah. Um, another, another part of, of pipelines and any production is, you know, keeping track of your assets, and version tracking, and stuff like that. What, what methods of version tracking have you found to be...effective, I guess?

I4: We used to version control the heck out of everything. We used to do very, very detailed version control. And, um, and expose all of that so that, uh, everyone had access to any version of any asset and, unfortunately that led to, uh...assets have a lot of moving parts. Assets and shots have a lot of moving parts, and, uh, you end up spending a lot of energy making sure that model a goes with rig b goes with, you

know, surfacing technique thr-, c, and that those are all in sync at the time it lands down in lighting, and, uh, and that they stay in sync. You know, that a new model doesn't come along and then now it's out of sync again, and suddenly a shot that the artist is in the middle of breaks down. Our current methodology, we've removed versioning from the artist. So we've moved closer to what's called a push mechanism where a new asset comes along, it gets put out in everybody's face. And, if it's broken, well, it breaks everybody. But, then it gets addressed. It's part of being a little lighter and more nimble, is, you will have more breakage but you will be on top of the breakage and you'll resolve it quicker.

B: Mhm.

I4: Um, we still do versioning, we just hide it from the artist. We have versioning under the hood so that we can revert to older versions of assets when something goes wrong, and for archival, you know, purposes, we can resolve what version of an asset was used for a shot that was rendered and finaled three months ago and we need to bring it back up for a, you know in the off case which does happen, where, and it's three months later and they want to do a publicity frame from that shot, and you bring that shot back up and it renders differently. Because the asset that's in the shot changed since it was finaled. And so you need to go, there's absolutely times you need to go back to an older version of an asset, um, to recover a given look or a given, you know, a given environment. And so we keep all of that versioning control, we just don't expose it. The artists just get the latest and greatest all the time.

B: Ok.

I4: Unless, unless we go and we expose it at the TD level. TDs can go in and wire them up by hand to an older version if we really needed to get a given shot out the door, but we've gotten rid of the nuanced version control and all the complexity that brought. Um, the, we got rid of all that being exposed to the artists just because

it was, it was too cumbersome. And that lead to a lot of heartache chasing down version mismatches.

B: Right. Um, what...in a production that may change over time, um, what kind of flexibility needs should be built in to a pipeline? Like if, um, your, like say they add more minutes onto the movie or something. What, what considerations do you need to build in to your pipeline to handle that kind of stuff.

I4: Um, usually that sort of thing doesn't need a lot of change to pipeline. Adding more minutes just means more shots, and adding more shots just means, you know, new assets will be coming along if the new shots involve those, uh, and then just, you know, uh, it's more about pipeline, from a production management standpoint, organizing artist, you know, man-weeks and things like that to figure out how to get the movie made. Uh, when they add, when they add more minutes that throws all of that into chaos for a moment while you try and figure out how you're going to accommodate that, and that in the end of the day, it, you know, it translates into more work hours which translates into either overtime for people or, uh, bringing in more bodies. And then that's a whole con-, a whole set of conversations that occurs, about how, whether, whether bringing people in to accommodate this is going to be helpful, right, because there's a training curve to get them up on the toolset. There's a productivity curve in getting them used to working with the people and the processes. Um, often, and especially if it's fairly late in the production, the answer ends up being that it's, it's actually better just to turn up the OT knob, uh, on the artists that already know what they're doing than to try and bring in more.

B: Right.

I4: So that's usually how minutes, uh, comes into things. But there's lots of other things, uh...absolutely we try to keep the pipeline nimble, absolutely we try to have a firm understanding of what we're expecting out of every department and in

what form we're expecting it, how the next department down is gonna be consuming it, um, just to understand the basic components so that we can be flexible, uh, when something changes. You know, when suddenly we go, 'let's get into, uh, captured camera work' instead of layout artists laying out cameras and now we're getting all our cameras from the live sessions in a lab and we come back with fifty different cameras, and we have to do camera editing and all that stuff, and that requires new tools to be written and all that stuff...and, that's an example that happened partway through Ralph, is they started really getting excited about the, the benefits of the camera capture lab that was being built. And they started getting excited about using it even though originally their show had no intentions of using it.

B: Right.

I4: And so, then, suddenly, a whole lot of stuff had to be written to accommodate that in their pipeline, accommodate a new set of data, a new delivery mechanism for the data. Um, so they adapted. They write, they wrote the tools and now future shows can leverage off those tools and improve them and polish them up a bit. And Family Movie is already using camera capture stuff for some of their work, and for the film after Family Movie we're already starting to talk about, in the next few months, getting into capture for all of that as well.

B: Cool. So I guess flexibility in a pipeline is less about the volume of work than it is about integrating new tools or work methods in? Because it sounds like you're just trying to, if you're getting more minutes out you're putting more, uh, down the same pipe, but you don't have to change the pipe.

I4: Usually, yeah. Usually it's just about productivity and usually you try your best to optimize the pipe to begin with, and so, by the time you're in the throes of shot production, there aren't really a lot of easy knobs to turn to become more productive at a given facet of the production pipeline, because if they existed you

would have turned them already.

B: Right.

I4: Uh, if you were aware of them, and then, you know. So we still look for those all the time, we're looking...there's an entire team on, on Funny Movie called the tactics team, and their whole job is to optimize, optimize, optimize. They look at render, you know, they look at render stats coming through the pipeline, they look at sets arriving. They look at everything. They look at rigs and they look at all these things and figure out 'how can we make those a little bit faster?' right, but they're doing that regardless of how many minutes are added, they're just doing that because we want, we already know the movie will be crunchy at the end, and so we want as much of that pain and anguish to end up on screen as possible, and so, that means making everything run as fast as it can so the artists can get the most out of their time at their desks.

B: Yeah.

I4: So we're already turning those knobs as much as we can find them and tune them. Uh, we're already looking for ways to optimize the production process all the time. And so, when they come along and they say, 'we're adding 6 minutes to the movie,' you just kinda go, 'oh, hell.' you know, and then you figure out, what does that mean, and it almost always is going to translate into more man-weeks one way or the other. Either more OT or new artists needing to be hired.

B: Yeah. But it's not something that would require you to rewrite some part of the pipeline.

I4: Ah, no, not...again, not unless those new minutes incorporated some new effect or something that you didn't have a way of doing, right? Which does sometimes happen, that some new idea will come along, not even necessarily new minutes added, but story rewrites. The story remains...one of the strengths of this place, as well as

one of the headaches, is that the story is flexible. They're always trying to hone the story to make the best movie they can, and so they can be relatively far into production when they pull the plug on something.

B: Right.

I4: And that is a recognized part of the Studio Delta way. And, uh, [other studios have] touted that as, they got that from Studio Delta, you know, that concept, it goes back to [the founder] himself. And around here, you know, back to the 2D days, they touted that a lot, you know. Traditionally Animated Film, there was a whole sequence where, uh, [the plot was different from the final film]. But they decided that wasn't playing well, and they wanted to bring [the hero in] earlier to have more breathing room to tell the [story] so that it, so that it felt less forced. Um, so they gutted it. They had already boarded, they had already drawn it, it was done.

B: Wow.

I4: They gutted that and re-drew it all and, and made it work with [the hero] instead and changed the story around in the final year of production. They did all that rework, and then it went on to be [very successful], and they, they point at that as one example of why this works, why constantly spending the energy to revisit your story and make sure you're telling the best story you can, and when you, when you can think of a better way of doing something, making the hard choice, you know, certainly the monetarily difficult choice of going back and revisiting it. Um, it's a really cool thing. You know, it's really cool that all the way to the finish line they're trying to always make the movie better.

B: Yeah.

I4: It absolutely translates into more work. You just, there's no, there's no ways about that. I mentioned earlier the notion of, you don't want to get started on your assets too early because they may end, you're, cause then you're just guessing.

They may not end up in the movie. Um, that happens all, all the way to the finish line. You can be a few months from done and they, and they have a revelation of a different way to do the opening sequence or something like that, and all of that hard work, some of which will already be animated and lit and completely finales and, and sealed up somewhere, uh, they could throw it all away and re, uh, and, and redo it all, just because it makes the movie better. And that's, that's a really cool thing, but it absolutely means a lot of work.

B: Sure. Well, um, you talked about transitioning from an old pipeline to a new pipeline. Um, who makes the decisions about how it should function and what problems it should address, like, what the new functionality should be. Is there like a, a top group of people that gets together and has, you know, meets about this kind of stuff and passes it down, I guess?

I4: At, at the end of the day the visual effects supervisor is the final say on everything that's happening on the floor for a given show, right? The visual effects supervisor's job is to interpret what the director and art director and, and character designer and all that want to make, movie-wise, and how are we going to actually accomplish that. So the visual effects supervisor runs the floor. Um, the, on our films here we have a technical supervisor that, that works alongside the visual effects supervisor, you know, for them as an advisor on the geeky stuff. Right, on, on things like process and pipeline and how, how new tools and things like the, new tools and evolution of tools and all that are going to help, help enable the director and art director's vision. So, the answer's kind of both. The, the VFX supe and the tech supe are a, are absolutely the, you know, eyeballs-deep in solving those problems, uh, figuring out how we're going to make the movie, including any pipeline changes that are necessary, new tools that need to be written, etc. etc.

B: Mhm. And does that go down all the way to the level of, like, picking naming

conventions for things, or...?

I4: Absolutely. I mean, there's a whole leadership team on a show, uh, all of whom report to the VFX supe. You know, sometimes through a couple of levels, uh, depending on the complexity of the given part of the process, uh, and there's actually flexibility in how that's structured movie to movie. Uh, the names of the supervisors and their areas, their spheres of influence will change movie to movie based on how the visual effects supervisor thinks this particular project might best be attacked, uh, or even the skill set of the people involved, right, based on who the visual effects supervisor's bringing on into leadership roles affects how they're structuring the leadership, uh, buckets.

B: Right.

I4: And that in turn adjusts the conversations about how those different buckets need to communicate with one another and how data needs to travel, and, which will eventually influence pipeline in various ways. So, um, naming conventions, all that sort of thing are settled, you know, either they come from the top down, the visual effects supervisor has a strong inclination of how they wan the show to function, or sometimes from the bottom up, you know, an individual artist or department head or something will say, 'we need to change this to make our lives better.' And they'll pass that up, and then see if that gets approved or not.

B: Right. [laughs] that's good. So there's, there's, can be input from everywhere.

I4: Everybody. Everybody has a voice, that's another one of the very many things I like about this place is, everybody has a voice. It doesn't mean you'll get listened to. I mean, you'll get listened to, absolutely. We always, we, we very much want to hear new ideas. We're always trying to reexamine how to make ourselves better, but the people, you know, and, an individual artist in an individual department doesn't have the full picture of everything that's involved in getting the movie made

in terms of where resources are and all that, so they could have a really cool idea and it may not end up getting made for that movie because, you know, the resources are all on more important things and we just don't have the bandwidth to tackle it, right?

B: Mhm.

I4: We write it down off to the side as a really cool idea, you know, we mark it down for something to work on for the future, but it may not get tackled for the movie they're on in that moment. Um, or, something they think's a really neat idea, sometimes even ideas that will even help a given department be more productive, uh, will cause hardship on other departments. There's cert-, you know, the flip side of that, there's certain tasks that are a pain in the butt for certain departments to do, but them doing that step makes other, several other departments' lives easier. And so, we make them do that step. We have conversations about ways to make it as, the, least amount of pain possible, but we're not gonna take that step away to make their lives better, because that's gonna make it harder on other people. And pipelines have a lot of moving parts, and, you know, not everyone on the floor in any given department has that full picture of how it all functions. And so, we're always looking for input on ways to improve it, but their input can sometimes be myopic. It can be focused on their little facet and on how to make that better, and that's not always going to be for the best.

B: Right. Uh, I guess I've just got, uh, one or two more things I'll throw at you from, from the other interviews. Um, uh, when I talked to Interview 1, he was talking about the concept of smart assets and having, incorporating more and more metadata into assets. Um, can you, talk about that at all? Is that something that you think people are moving towards, or...?

I4: Um...oh yeah, no no no. And, we absolutely do tons of it. You know, all

kinds of things. Um, you know, the more the context the asset was created in and the intent of the artist, the more you can capture that, uh, the better able you are to understand how it needs to be used correctly. And so we use a lot of metadata just to store attributes on things and carry along, just, pieces of, the snippets of data. Um, either from artist's intent, or even just, uh, production byproducts. You know, for instance we have a, we have a little attribute that gets tagged on, uh, cloth meshes that have been up-rezzed. You know, when they're in the cloth department they up-rez the mesh to,uh, you know, refine it so there's more control vertices in the mesh so that the cloth sim can generate more smooth motion and whatever, they will up-rez that mesh and it gets heavier. Um, that mesh then gets passed downstream. That mesh no longer matches the one in the stockroom. It no longer matches the original asset, but there's a correspondence. And so, that's a little piece of data they, a little piece of metadata that they tag and say, 'oh, by the way I up-rezzed this twice.' and then when it arrives in lighting and they start trying to stitch it all together to make a render, they go, 'okay, go grab the look from the stockroom, and wait a minute, this doesn't match, but oh! This attribute's here, they up-rezzed this twice. Okay, so if I up-rez this twice in the surfaces, and then we know with [our systems] and everything how that all aligns, then, ok yay! Now it matches. Plop. Render. Done.'

B: Yeah.

I4: Uh, with little things like that, even, are just purely a part of the production process, metadata gets tagged onto it to help communication, right? Just to help pass that information along, uh, without them having to pick up the phone and go, 'what the hell were you thinking when you did this?' you know, it, it's to help describe intent a lot of the time. So, yeah we use it all over the place for little, little things and big things.

B: Mhm. Well, uh, one thing that Interview 2 talked about was, uh, moving

towards - and I think this is more of a, like, pie-in-the-sky note, more than anything else - um, moving towards kind of a check-in, check out philosophy with assets?

I4: Uh-huh. Yep.

B: Um, you know, with an asset they'll grab something, they'll do something with it and they'll put it back in, and...

I4: That's very much the way we're working, it's a hub-and-spoke data model. Yeah. We used to work purely assembly-line. Uh, a, a given piece of data was created and then passed to the next person, and they did stuff with it and passed to the next person, and so on. Um, and then when it went back, it became a challenge to figure out how to do things, and, you know, and, where the data was living in any given moment, and the state of the data was funky, and, and sometimes because of that data, and because of the localization aspect, sometimes problems are getting solved in different ways, and if they're not, if they're not doing a good enough job talking to each other then the data goes through state transitions as it's passing along. And the data in the downstream department may not even have the same form as it did in an upstream department. There are strong aspects to the production process that benefit from the assembly line. The benefit from that passage of a given shot through departments. Um, sometimes it needs to move back, though, for a variety of issues, you know, quality control kinds of issues just like that sometimes happens on an assembly line. The difference is that we were using that model at the time both to describe the flow of a shot through a department and the departmental structures, but also the flow of data on disk and how it was all stored and accessed and all of that. It was all following that same paradigm.

B: Mhm.

I4: And eventually we just said, 'they don't have to be coupled.' and so, even though department A will work on something, then department B, then department

C, then department d usually, um, we've changed it to the data model under the hood being a hub-and-spoke model. So department A checks out the data, makes their changes, checks it back in. Department B checks out the data, makes their changes, checks it back in. Oh wait, department A needs to do a little more work - they just check it out, do some more work, check it back in, and so on. And so it makes it more flexible for departments to work out of order, uh, while still the normal workflow begin assembly line. A, an assembly line style of work. But the data under the hood is all check-in, check-out.

B: They're not necessarily getting it from the upstream department, they're getting it from...

I4: Right. It, it's not passing the baton from department to department like a relay, it's, yeah, it's going and checking data out of a central hub. So it's a hub-and-spoke data model. Um, it's been very nice for us. We've done, we moved to that for our data model about two years ago and, um, even though for the most part the shows are still working assembly-line, it's made the exceptional cases considerably easier, uh, and it's enabled certain things like auto renders. You know, the notion in that, because we've moved to a more common storage mechanism, even, for the kinds of data, we got, we squashed out state transitions almost entirely. And so, uh, as soon as layout pushes out a version of the shot with really rough models, with the really, you know, the beginnings of the camera move, with characters kind of, really really really roughly animated because the animators haven't even touched them yet, just the layout artists did some basic posing to kind of suggest intent in given shots, it's renderable, right? Everything's renderable from the very beginning. And so we've used that, uh, to enable what we call auto renders, which is, any step of the way, any state transition, from a production process standpoint, a new, a da-, a piece of data changes, a given asset changes, uh, layout revisits the camera, you know, any

department makes a change - animation pushes out their rough blocking - anything, uh, tells the auto render toolset that something has changed about this shot. And they go off and re-render it. So, a new version of the primary character comes along and they push out a new version of, [hero], let's say on Funny Movie, or [sidekick] - half the movie re-renders, because they're in so many shots.

B: Right, yeah.

I4: So, uh, so hundreds, sometimes a couple thousand renders go out on the queue at night, uh, at very low priority to re-render the entire movie, and we use that to sanity check the assets.

B: That's crazy.

I4: So if some change happen to an asset that makes that asset heavier in some regard - take more memory, take more time, whatever - uh, we catch it very quickly now, because it, we, we see it in a whole bunch of shots often before artists have even had a chance to pick it up. So it's been very very nice. And the hub-and-spoke data model helped make that easier, because everything gets checked in in a renderable state.

B: Ok. Well that makes sense. I can imagine that would help a lot, actually. [laughs] um, I guess I just have one more question. Uh, when I was talking to, to Interview 3 at TV Studio One I asked him about command-line and GUI stuff. Um, they, they don't use any command line stuff at TV Studio One. It's all, it's all GUIs. Um, and he said that, you know, the time where it was difficult and cumbersome to write GUI interfaces has passed. And like, there's no, there's no need to force an artist to use the command line. So, what...do artists at Studio Delta have to use the command line? Are there GUIs for things? Is it a combination...?

I4: It's still a combination. For the most part, uh, they have UIs for most everything nowadays at the artist level. We build command-line tools more for TDs

than anything else because there's a lot of times, absolutely you can build a UI, but the UI is cumbersome in and of itself. It's sometimes easier to just go, uh, this, on really flexible components of the pipeline, and I want to do this publish with just the camera, and I want to do this and I want to name it foo, and blah blah blah...it's sometimes just fast to just type it. And, and so we have command-line interfaces for a lot of things. Mostly those are for TDs. It, it's relatively rare that artists are touching the command line anymore. It's very much artist tools, you know, layered on top of maya for the most part. We use maya as an application framework and build tons of tools on top of there. Um, and yeah, it's relatively rare that they have to go down to the command line for anything.

B: And that's just, sort of, to, so they don't have to worry about that kind of thing, right?

I4: Yeah. Yeah, I mean, it's, it's...it enables, it enables a certain level of ease of use, uh, through the UI. It also constrains them in some regards. Uh, they, we are controlling the access they have to data and the state they get the data in, because they have these set of buttons to press to get the thing they want. Um, whereas, when it was more command-line in the past, they had a lot of, they could do a lot on their own and they could get themselves into trouble if they didn't understand what they were doing.

B: Sure.

I4: Um, UIs often make that a little easier because it's, it contains the artist's experience a little bit as well as enables the artist to experience. It makes it friendlier and easier for them to do the things we expect them to do, it keeps them from doing the things we don't expect them to do. So, it's good for both of those things.

B: Okay. Cool, well that's all I have so thanks very much for your time. I appreciate it.

I4: Cool. Absolutely!

B: And hopefully I'll talk to you again.

I4: Awesome.

INTERVIEW 5

BRANDON: Hey Interview 5, it's Brandon.

INTERVIEW 5: Hey Brandon, how are you?

B: Doing well. How's it going?

I5: I'm going–it's going alright. Sorry we had to move this a few times, but...

B: No, it's okay. Uh, I understand. Meetings can [laughs] can–can happen all of a sudden, so...

I5: Yeah especially–yeah, when I looked at my calendar earlier in the week and Friday was wide open, I though ah, this is too good to be true...but, um. So yeah, I'm glad we're able to still find time.

B: Great, great. Well, thanks for taking some time to talk to me–I–I appreciate it.

I5: Sure.

B: Um, did you have a chance to look over the questions list I sent you?

I5: Yeah, I did. Yeah, yeah, they look–they look good and I actually have them up here...

B: Oh, okay, cool.

I5: Uh, just to cheat a little bit. [laughs]

[discussion of recent internship, not relevant to the research]

I5: So yeah, we can start whenever you want to start.

B: Very cool. Um, I guess we'll start with the background portion. You know, kinda talk about uh, your current position, how long you've been there...kind of some things that you've done. How you got to the–to–to–to Studio Alpha and kind of the things you've been working on. Talk about that first.

I5: Sure. Okay, so I'm um, currently a character TD, um, at Studio Alpha. I'm,

um, right now I'm on what's called global development which means, um, that I'm sort of developing tools and workflows and um, that support all of the shows and then also looking into tools and technology um, either developed internally or externally that we can implement, um, in the future to support, um, future shows. Things that, you know, tools, technology related to character setups that, um, we might need in the future–let's say before–that's why I've been doing–that's why I've been on global development for just a month. Before that, I spent 18 months on uh, setting up characters on [a movie], um, which comes out next year. Um, and actually, that's right around then, so I've been back at Studio Alpha for um, a little over–a little over 18 months.

B: Okay.

I5: Um, I had worked uh, at Studio Alpha in the [early 00s], on things like Film A and Film B and um, the [theme park] ride. Um, and then between those times, I actually went back to Metro City where I'm from and did a variety of things. I wrote a book, um, I um started my own studio. Um, I freelanced around at different studios before that. I also did some teaching. I ran a research lab at University One, um, so... yeah. Um–just to go back a little further in case it's interesting... my undergraduate degree is in biological anthropology.

B: Oh that is interesting. [laughs]

I5: Well I studied, uh, sorry, my phone is ringing, but it's not–not important. Uh, so I studied biological anthropology–essentially um, human anatomy and human evolution, um, from like a really functional point of view. Um, and so I did that for a long time–I actually started in a phd program in evolutionary anatomy and uh, decided that I actually wanted to go to art school. So...[laughs]...so from there, I went to Design School One and I got an MFA in uh, program there that they have called Design and Technology–which is kind of like the–the Viz Lab, but, uh, maybe

more geared towards the sort of design world, but you know, like heavy programming, heavy animation. Um, technical, but–but always in the service of your design goals. So and then you know I worked a little bit around Metro City for about a year and then I went off to Studio Alpha right after grad school–probably less than a year after I graduated. Yeah, so, yeah sorry that's a–sort of a–like non-linear non–

B: No, it's okay. It's nice to get a get an idea of where everyone's coming from, get a different idea of their experiences and stuff like that.

I5: Yeah, yeah. Absolutely.

B: I've talked to some interesting folks, so...

I5: I'm sure–yeah, I'm looking forward to reading your–your thesis when it's all done.

B: Thanks–me too! [laughs] Um, okay. I guess the next thing talk about, um, talk to me a little about openPipeline–like what it is, um, maybe the motivation behind it, um–and I know a lot of this is covered in the literature that I've read and I know in some talks that you've given, but just kind of briefly, just so that I have it in this–this interview document, um, kind of talk about what openPipeline is.

I5: Sure. So openPipeline, um, was my attempt to uh, create an open source um specification for how an animation/visual effects pipeline could be organized. Um, it–the motivation for it was that I had been freelancing around in Metro City and–and um had been exposed to different pipelines and I kept on um, being tasked with the idea of creating a pipeline tool, because most studios in Metro City don't–or at least years ago, did not–have pipelines. The pipeline was "don't save this on your desktop, save it on this server." Sitting here in the corner and put–ideally put it in this project directory, um, and ideally name it something useful. Um, and just yell out when it's there. [laughs] basically.

B: [laughs]

I5: Um, so there was no real sense of structure–there was no real sense of–um, you know–organization. And um, so then I landed, um, I spent a about a year at a studio called Small Studio One, um which doesn't really have a studio anymore, but it's an art and technology center in Metro City. Part–part of its old structure was that they had this production lab where uh, we would–where the lab would essentially do visual effects and animation for fine art projects. So things that weren't like necessarily heavily funded and they weren't um, headed necessarily to your standard, uh, visual effects/animation/games sort of delivery platform. So, um, they were going into galleries, they were going into museums, they were indie indie films that were being created by artists, and so this was kind of like an experimental center for–at the time, you know, heavily into doing visual effects and animation, and even a little bit of interactive programming.

B: Right.

I5: And so I was the, um, studio technical director there. And um, they were also really into open source software. So I said well, um, you know so I'm going to start writing this tool to manage these projects and I'd like to make it open source. They were like that's great. Let's go for it–because they were a non-profit. So they were really into the idea of sharing and building community around things. And so it started there and then um, we–I was–I'd been adjucting–adjunct teaching at University One on the side and all of a sudden, they decided to open a research lab and they hired me as one of the first researchers. And I said well, I'm going to take this project and keep working on it there. And so then it became something that I really was creating for my students too, because I realized, okay, that I hadn't taught in a while and I'm a pipeline nerd... and I'm pipeline obsessive...and I just felt like it had saved me and I wish I had had it in school and so I started developing it with my students in mind. And so it sort of took on there–I also started working with–I

had a grad um, I had a graduate research assistant who worked on it with me.

B: Mmhmm.

I5: And I–and we started putting it out–we started putting out versions and then we started realizing that okay, we thought it would just kinda be for really, really small indie projects or for students. And that would have been fine and then it started to build momentum when we found out lots of studios were using it. And before we knew it, we realized that many studios, particularly in Metro City, were using it because it was word of mouth. Um, and so it really, then, took off. Studios started to contribute to it, and it reached this you know almost 1.0 state and um, it just sort of reached this point of stability that got really nice. There's still like always little bugs and they're I have a long list of feature requests that either I've, um, put together or people have sent me. Um, but the goal now for it, eh, um, it got used–it gets used quite a bit. Like eh–I, you know, I, you know I'll go to studios I've never heard of, walk in, and all the artists there are using it. Um, and, um, people have extended it in some ways. Like there are some studios that have done like some really interesting–like there are some studios that have done like have created some really interesting custom versions that they don't reintroduce back into the code base...

B: Right.

I5: Um, but you know it sort of reached this point where the–because it had been developed in this pre-Python uh, inside of Maya era, um, it's just this massive MEL behemoth. So the idea was we were going to stop development on it in terms of MEL and maybe just do both fixes as possible. But to really completely re-architect it in Python because um, production pipelines are inherently an object oriented problem, so...it's um–It would just be much more suited for that and it would just be so much easier. And as we started, we realized that we would be able to create all the same functionality with more and it would probably be like a quarter of the size in terms

of actual code. Um, so yeah, that's kinda where it's at–and it's out there and people still use it and um, sort of unofficially, I've heard that the folks at Tank–um, they–there's an asset manager called Tank–um, and it's created by the same folks that make um,–what the heck's it called–Shotgun?

B: Oh, yeah, Shotgun.

I5: Shotgun. So Tank is their asset manager and–and at SIGGRAPH, you know, I didn't hear this–I mean, I spoke to–to the creator of it directly. And without really saying it, he kinda insinuated that they had pretty much based a lot of the ideas of Tank on–on openPipeline.

B: Oh wow. That's cool.

I5: [laughs] Yeah, so that's interesting. And I took it as a huge compliment and uh, so yeah–so that's kinda the history and the motivations for it.

B: Yeah, that's great. Um, that's all good to understand as we will discuss kinda some more things about pipelines it kinda helps frame that stuff out.

I5: Great, great.

B: Um, okay let's launch into the other questions then. Um, we'll start off with the big one–in your experience, how would you define a pipeline?

I5: So–so a pipeline is–is two things. Um, I always consider it um, both the ecosystem in which the production, uh, lives in um, and it's also the culture of the production and by culture I mean sort of the rules for behavior. Right? It's–it's it dictates um, how the project is done and it answers the questions, um, that the–it essentially–it answers the questions that the production has sort of come up with. All the questions that they've raised–the answers then become part of the pipeline. And so that they don't have to be answered again. Um, and by ecosystem, I just mean that it's–it's creating the structure and framework, um, and interdependencies and connections, um, between all of the assets and the relationships to shots and um,

you know, basically everything from art to output. Um, so and then with regard to culture, it's–it's really about like where do files live? What do I name them? You know, all those kinds of things that you need to know in order to understand how to speak the language of that production, um, and to relate to it on a day-to-day basis. So, in a very–that's my sort of like [laughs] philosophical description of what a pipeline is, and–and from there, you know, everything varies widely. But–but to me that–that is what a production pipeline is.

B: So what–what characterizes then a good pipeline?

I5: Um, something that's really, really clear. Um, something that's–it's very well described and thought out. Um, I think that it can take many, many shapes, and it really depends on what works for the studio, but a good production pipeline um, essentially handles your version control, asset management, um, artist communication, um, you know all of those things, um, basically without being confusing. Right? It's intuitive, um, and–and a lot of this is insinuating that there's a tool–let's say that's used with the production pipeline and I don't think that necessarily has to be the case–as long as um, you know the rules are very, very clear about where things–about where things go.

B: So then, um, you mentioned some things about asset management, um, version tracking, communication tools, what–what kinds of things–what kinds of–well, let's start with communication. What kinds of communication tools have you found to be effective–both in your experience at Studio Alpha and, you know, looking at how people use openPipeline?

I5: Sure. Um, so I think, you know, artists notes on an asset are incredi–are incredibly important. You know, just being able to say this is what I did when it was saved. So that when you–you really just understand that that object has a history and you can travel back and read its history. Um, so that as a very basic thing is

good because lots of times, a production pipeline is a-synchronous with regard to the users actually interacting with it, right? It's not a–it's not a thing that everyone is using at the very same moment and there's opportunities for chat um, and even um, just talking, right? Just uh, you could be working on a project here with a staff in India–12 hours difference and you're never on the box at the same time. Um, so being able to really associate information with–with revisions is really just a basic thing that needs to be there. Um, I think, you know, any time that you can just provide people with a place to just sort of write notes in a freeform way that is not related to revision I think is also really, really useful. So we ended up adding, you know, a pipeline sort of like notes category so you could say here are the things I think need to be done–um, or, you know, new model is coming soon–keep an eye out. Um, and then triggering these things with email, I think, is always important.

B: Right.

I5: Um, you know, and just being able to–as something's changed, um, or someone makes a comment on something, then you're able to just get and email just in your inbox and it just sort of triggers you to–to sort of look at it. Um, I think those are useful. I have this challenge at one point to–for people here to describe our production pipeline without ever using the word email. And people really couldn't. That's still–that's still like a problem–I think there needs to be something that goes beyond that.

B: [laughs] Yeah.

I5: Um, and so, I think that, uh, you know, that we'll see how that sort of progresses over time. But you know, I envision a time when there is sort of an integrated production operating framework that you're working in. That everything else sits inside of–like your Maya session sits inside of and all these other things. Um, and so that you're actually able to like see–see things happening and just see live

changes and–and just see all of this, you know, almost the way a day trader on the stock exchange is gonna be able to see a huge amount of data going by...like I would love for us to have a tool that just allows us to see everything that's going on–when you want to, you know, or...

B: Rather than a–getting an email notification about something.

I5: Yeah–exactly. Hey this file changed. Go–go check it out. Like you know and then you have to stop everything you're doing and–and–and uh, you know, update or do whatever it is–whatever the action is that's associated with receiving that email–whether it's automatic or manual. Um, yeah. Kinda, kinda tricky.

B: [laughs] yeah, I–several other people that I mentioned kinda mentioned the–If not the need, but definitely the desire for something other than email just because it's inherently noisy and um, yeah. It's just not ideal by any means. And uh, I talked to David over at South Park Studios and he said that–they're obviously a very small studio there, but, you know, he said that they usually just shout at the guy, you know, and kind of communicate things face to face as much as possible just because they can sort of more clearly convey what they need to–

I5: Oh yeah, when that's possible, it definitely helps. I mean, when we had Small Studio Two in Metro City, we were small enough that yeah, we could–you know we had this very strong infrastructure/production pipeline, but yeah, it was just super easy to say okay, file saved–go get it. Yeah.

B: Yeah. Um, talk a little about feedback loops in a pipeline. Um, what–first of all what is a feedback loop and how–how are they used and um, I guess if you can provide and example of something like that that exists in openPipeline, um, and talk about that as well.

I5: Yeah, I don't—in terms of openPipeline, I don't think there's really the–the–the true sense of the feedback loop because there's no real way that the client

inter–ever interjects with something like openPipeline. Um, but, we–we have–like at Small Studio Two, we did develop other systems that were sort of uh, more sort of client centric and so clients could sort of do draw overs on, you know, a thumbnail or a output image and then feed it back into the system. Um, and then that would become part of the–the asset history for that. It's like here's a revision note, here's a series of notes related to this asset and you would have comments, you would have revisions, you might have snippets of emails, um. You know, whatever it–whatever it took. And I think that can be very, very valuable. And we do have some systems her for things like that, but uh, I think there's still yet to be, that I've seen, like the–the end-all be-all kinda of like visual feedback system, um, that you can relate to a client. And that it's easy for them to use and it–and on your side, it really interfaces with your assets, right? Because your clients don't necessarily need your assets–they just need what becomes of your assets, whatever that may be.

B: Mmhmm.

I5: Um, so, yeah, I think, you know, but I think that there's–there's a lot of need for that. And it's good, especially on long-term, large productions, it's really good to just have like a long–you know, however big that history ends up being, it's good to have all of that trackable. Because then you can really see how long things took and then it's like oh wait–this took–you know, doing an asset of this style or of this type, you know, ended up taking us a lot longer than we thought. And here's why–here's the same feedback on all these different things. And so from an asset point of view, I think feedback loops, um, and feedback documentation is really, really useful and hard to wrangle because it, a lot of times, comes in such disparate forms. It's like oh, here's a chat session. Here's an email. Here's when a guy walked over to the desk and drew on a piece of paper. And like, you're getting feedback in so many different ways, um, that it would be good to have all of that logged into some kind of catch-all

system, especially if the whole thing was digital to begin with. But, uh, but yeah, no–never really had that sense–It was just like here are my 800 revisions to get to this file, and I can go back and get any of them, but, um, yeah, it was sort of a closed system for internal production as opposed to something that has like uh, output to a client for feedback.

B: Right. So like a–a–say if a student were using it, they would sort of be their own internal feedback loop. Like–I like this, so I'm going to publish it.

I5: Yeah. Yeah, or here's the notes I got from my class or my professor when I showed it in class yesterday, and you know, the next, you know version reflects that in some way. But–but it's not all that information isn't inherently inside of that system, um, because it is meant to just just simple. You know, like, but–that's something that' I've always sort of thought about–that it would be nice if you could reference in other notes and other forms of data.

B: Um, talk a little bit about the version tracking solutions that you have as part of openPipeline and um, as well as um, I guess any other–any other thoughts you may have about version tracking.

I5: Sure, yeah. So openPipeline went for the simplest possible solution which was just to–a–everything you made a new version, it–it put a copy of that file in the correct place, named appropriately for that new version. So you never, ever, ah, hit "save as" and then named it something. You should basically save a new version, it would put it in the right place, if you double clicked the asset, it would open the latest version and you would go from there. The reason for that was that we wanted it to be really, really simple. And all that it needed to do was you just pointed it to a root directory and it handled all the directory structure underneath for you. So a lot of times, an artists didn't even need to look into where the directory structure was. They just opened the asset and made their changes and, you know, saved the version

or mastered it, um, which would create a published version and they–and they would just sort of um, go from there. And, you know, of course, you're taking–you're making the assumption that disk space is cheap, um, at that point, um, which generally it is, um, but we did–we did have to put in–and this was like feature requests that were put in–we had to put in um, functionality to just call old versions, right? Um, so you're just sort of getting–getting rid of–of just save the last ten versions–and that usually was fine because for the most part, you know, you had enough backup or things really hadn't changed so dramatically you needed to go back to some previous version.

B: Right.

I5: You considered every previous version to be–to be, um, revision history or I should say like, yeah, just history on the final object as opposed to oh here's a whole new character. So sometimes when characters when through a big um, ah, design change on projects that were using openPipeline, they would just create a new version–a new asset, or something slightly different and they would just start from there, so, you know, it's just everyone kinda does these things differently, especially if it was so dramatic that it didn't look anything like the original–the original character design. Um, and so, you know, but in other cases, you know, I've–I've y–Implemented and been exposed to more heavy duty version control–things like Subversion or Git–other, uh, systems that are–are not um, sort of duplicated based, but are repository based and basically you're just saving differences between files. Um, at that tends to be nice and I have done whole like, you know, 30-second commercial spots, uh, using a web based SVN server. And that's been kind of interesting.

B: Yeah.

I5: Um, especially if we were were working for a client or delivering assets or, you know, we had a couple of people working off-site. Um, that was kind of nice because

we could just check-in or out files and that actually worked pretty well–better than I thought it would. But it wasn't a suit–the models weren't like giant Zbrush files–like everything–we were able to keep things like really, really modular and small and uh, yeah and we pushed binary into SVN and it a–it just worked. Uh, I wouldn't recommend it for everything, but, you know, this sort of cloud-based idea ended up working pretty well, um, for certain projects. Um, certainly here at Studio Alpha, we have our own revision control system and–and a few more in development and um, so I mean there's just a long tradition here of–of solid revision control systems.

B: Mmhmm. Yeah, one thing that I remember Interview 1 talking about was, you know, maybe the situation where a director says, I want um, this version of say Hero, like from the first trailer that was released, and by that time, Hero has iterated three or four times and doesn't look that way anymore. And so, um, you'll be able to go and grab those things, you know, t–um, for whatever reason they may have a marketing shot or whatever.

I5: Right. Yeah and yeah, that's kind of–we can get back to previous versions of characters, um, but that doesn't necessarily–for us it's the kinda thing that–um, it definitely takes a little bit of work. There's interdependencies with shared resources really, with the characters. So it's not always like oh yeah, we're working with version 80 of this character. Let's just go back to 50 and expect it to work. Um, a lot of times, there's–there's a few interdependencies that are related to those things.

B: Right. Um, what kind of flexibility do you need in a pipeline to handle different project requirements?

I5: Yeah, you know–this is definitely a tricky thing because that's–that's one of the hardest things about openPipeline–Is everyone wants flexibility. Um, everyone wants it to work in their particular way. And we've definitely said no to people at times just because they want it to work in a completely different way than it already

does. And so I think that like, flexibility in a pipeline is something that only can go so far, right? It's only the kinda thing that, you know, the pipeline needs to inherently be rigid. It needs to like, have some basic rules. Um, and then, you know, flexibility has to come in sort of smart and small ways, right? It's like oh we need to override something in a shot or we need to override something um, you know, on a particular case for this character, only when it's in this variance state or something like that. And so I think the flexibility is one of these things that–it just–I would say that once the production is rolling, you really can't have a whole lot of flexibility.

B: Mmhmm.

I5: Um, you know and of course like once it gets into lighting and stuff like that, you know, they'll do whatever it takes to get the shot. But up til that point, like, ideally, you want it to be as–I would say inflexible only to–to emphasize the word consistent, right? Because every–every asset needs to behave the same way, every shot needs to be set up in a very, very similar way so that anyone, you know, we have hundreds of artists here all working on the same movie, so that anyone needs to be able to open a shot and say oh, yeah. Right. This is why this isn't working. Not like oh, let's see what's custom is–what's customization here that I need to sort of strip out so it behaves like all the other shots. Um, so I think flexibility is one of those things that, um, I think is sort of more like a long term, sort of evolving thing, where it's like oh you know on that show they did it that way, but on this show, we want to do it a little bit differently. Um, or we want to build in for flexibility here and let's put in a hook at this point so that we can branch in some way and have a little bit of flexibility. Um, but I think that has to be done like–in a very sort of like careful and considerate way. And I think the point of the pipeline is just–is again to just set up the rules for how this is going to work and then ideally enforce them. Um, so I think pipeline and flexibility–well, I think there's definitely room for them to coexist.

I think it's always–it has to be done in small ways.

B: Yeah. Well like you're not gonna try to integrate a new software package halfway through a show or something like that.

I5: Right. Even if everyone wants it, you know, it's a really hard thing to–to do. Yeah.

B: But I think there is some–If I remember correctly, there's like some XML based stuff that–that is part of openPipeline that creates some data that I guess can be grabbed by, you know, not just Maya, it might be some other package or something that's...

I5: Yeah. Yeah, absolutely. And so, so I think, you know, flexibility can come from data access. I think that's a really important part of–of what openPipeline does which is to say let's put as much of this data outside of–of the sort of standard binary or Maya specific file formats and let's put things into XML wherever possible so that you can access it from the outside. Um, yeah, absolutely. And you know, there's a preferences page in openPipeline that used to be like there were two options and now it's like as big as the whole UI there are so many different things that people want to configure. Um, so I think, you know, we've definitely tried to make it customizable, in a way. Um, but we've always sort of stressed that once you've customized it for a project, you should basically leave it as-is for the extent of that project. Then on the next project if you want to change it, then, go for it, but you know, it's kind of like, each project needs to, you know, each projects needs to have it's own sense of rules.

B: Right. Um, so who–who decides–who makes those decisions about, you know, file naming and–and directory structures and what–what the parts that make up the pipeline? Like who makes those decisions?

I5: Sure. It's definitely a consensus thing. I would say that there's, meetings upon meetings and layered sub-meetings upon sub meetings and [laughs] higher level

executive conversations, and it's sort of–at a large studio, as I'm sure you've seen, it's not something that changes quickly. And so, a lot of times, this is oh, we're talking about changing the pipeline for a movie that is, you know, five movies out or something. Because you don't necessarily want to break everything that's been put in place already. Um, so I would say consensus by far–you know, that comes from the td's and even higher level than that, um, and then all the sort of relevant departments need to sort of vet these things. And make sure that it works for them because sometimes each department needs to change their code base in order to work with it. Yeah, so it's yeah, consensus for sure.

B: Yeah, that makes sense because if you had, you know, individual departments saying it should be this way or this way, they may not, necessarily, have the global view of what needs to happen. And...

I5: Yeah, yeah. And you can assume we're the only people or we know, you know, our downstream department and we know what, you know, our upstream department. But you don't always know, like who's kind of like tapping into your code, or tapping into your shots or setups in some weird way, you know, that you never anticipate. Like I've just been learning how lighting sort of taps into our setups a lot and the things that they actually need out of our setups, um, that we would never even think about. Like we expect them to just be looking at our models that we're churning out, but there is some aspects of setup that they'll need to look at at times.

B: Um, okay. Then I guess we only have a couple more things to–to cover. Um, how has–from the beginning of openPipeline till now, how has the project evolved–how as the um, I guess the main structure and the kinda purpose of it hasn't changed necessarily–but, how would you say the project has evolved over however many years? It's I guess like six or seven years at this point?

I5: Yeah. Just about, yeah. I would say that the flexibility is where it's evolved. It's that you know, it's where it's provided lots of different options for how people want to do things like, you know, what file format do they want to save files in, what's the path, you know, what's the path? Um, what do you want to call these paths and that sort of thing? Um, I would say that that's where a lot of the evolution has really come from. Um, I think in the beginning it was basically just like a glorified asset browser that you could just like, you know, here's where my stuff is, I want to open it. And then we sort of integrated the shots and sequences and stuff like that into it. And um, so, I would say yeah, I think the core mission remains in tact. And one thing that–I–you know, it started as a specification for a pipeline. The MEL version in Maya was just supposed to be an example of that. [laughs] And so that we have heard about people building um, Nuke versions, building After Effects versions, building Blender versions based on the specifications. I've never seen any of them, but I've heard that they're out there or studios have made them. Um, and so, yeah, I mean the initial idea was just this specification and then over time, it sort of became synonymous with this Maya version. And hopefully, you know, hopefully, it keeps going. I know that there are people who are working on, you know I don't really have tons of time these days to work on the Py–Python version, but I have heard that there is some–some activity on there. I don't know if it's making its way into the Google Code repository, but, um...you know, the last time I checked on it was before I came back to Studio Alpha.

B: Yeah, yeah.

I5: But, yeah, who knows. And we'll see kinda of how it goes, but I would say that flexibility and customization has been where its really um, where it's really ended up where I didn't expect. In the beginning, I just wanted it to be like this is my world view of how a pipeline should work and everyone–whoever wants to adopt

it, adopt it and that's it. Like this is going to be what it is and then it definitely became more of the sort of let's make all of these potential users like, happy, you know. Um, we definitely went through about six months of that.

B: Um, how–I guess this is sort of the last point–and how do you think a collaborative open source environment, um, not necessarily an academic–well, yeah talk a little about how the academic project environment differs from um, a commercial production environment at a sm–maybe a small studio, versus say somewhere like, Studio Alpha, that's big and does sort of feature films. How do those things differ and what kind of challenges are kinda presented in each of those environments?

I5: Yeah, I think they–they differ in many ways based on their sense of urgency. You know, a small studio, a small commercial studio is looking for solutions immediately, right? Um, I think an academic environment is looking for solutions maybe? Right? They're like yeah, we could–that–it'd be cool if it did that. But like, we're not really expecting anything and we're certainly not willing to pay. So, if it could be there, that'd be great... and I think a large studio is looking for solutions soon. Like, you know, whenever, you know, as soon as possible. Um, and so I think, to me, having worked in all these settings, I think that that's kind of–the sort of–um, the main difference, um, you know I think when you're building something that's open source, then you're trying to make lots of different people in different scenarios happy. Um, I think when you're at a stu–a large studio, you're sort of trying to make something that's consistent and long term, right? It's something that's going to last for a long term–and doesn't necessarily have to be too flexible is because all the shows are going to work similarly. Um, and then when you're dealing with, um, you know, like again an academics scenario where you're sort of building a tool for sort of academic use, they're sort of just happy to have anything. [laughs] Whatever exists. And so yeah, I think that the others, there's all of this kind of like little interrelated, um, sort of

sets of needs, um, that I think are all, yeah, I think that they're probably all very similar, um, in their tone. But I would say that's really the major thing. Um, and I think, you know, the open source thing was definitely interesting because at first, you put something out and make it for free and everyone's like hey this is so cool–this is great. Hey–It would be great if it did this. That would be great, you know, if you get the chance, and if I get the time, I'll try to put it in myself and I'll let you guys know.

B: Yeah.

I5: That's kinda how it always starts. And then later on, you definitely have studios who will write you and say, "openPipeline is broken. You–you need to tell us how we're going to fix it." [laughs] "Like, right now." That's like wow, okay. And you know, I would really try whenever possible. And a lot of times it was just like, you know, a user deleted the XML file, or a user just hand-saved a file in a different place and that's why it's not showing up. I mean, it's–I never encountered a situation–and luckily, like knock on wood, considering how many lines of code it is, where it would just went out there and it was just completely broken, or it was something that we completely missed or, you know, and so we had to do like a lot of like super safety things like we made this decision early on that openPipeline would never delete a file. It didn't have the ability. It would move it, fine, but it would never delete it, because we didn't want to be responsible for that. [laughs]

B: [laughs] well that makes sense.

I5: And so yeah, it's just like all these things and like you like start to take on like the responsibility of other people's projects that you have no–no knowledge of some Romanian studio making, you know, a full American TV show is like using this thing. And you're sort of like on call for no pay and no responsibility. So yeah, it's just sort of a weird thing that starts to like take off. And as soon as you're like

hey–you know–is there kind of a freelance. It sort of sometimes the support would turn into a situation where we'd say, you know, is this something that you want to contract us for. You're asking for some sort of a new feature that we can add in, you know, just for you guys and of course, they like disappear. [laughs]

B: [laughs] Of course.

I5: But yeah, it's kind of an interesting thing. It definitely–what I loved about openPipeline is that it got me, um, exposure into lots of different pipelines and like getting to talk to people and like even at SIGGRAPH, like I would schedule time to like meet up with different studios that were using it and just like pick their brains and introduce them to other studios that were using it, and building this sort of like ad hoc user group, um, just so that they knew each other and they could support each other, as well, but it yeah, it definitely–definitely got me involved in projects that I would not have seen had I not made it, so. Yeah, that was kinda–I don't–yeah, that was–I didn't expect it. I guess I should just say that was an unexpected benefit, you know. I thought it was just going to be this little quite thing that would go out into the world and people would use, maybe. Um, but it ended up being something that sort of introduced me to a lot–to a lot of people. Like I even went to Singapore for a week, um, just to do like some lectures and stuff like that. And I walked into a studio there of a friend–kind of an internet friend–someone that, um, I knew through pipeline had developed a bunch of stuff for openPipeline. And he took me to like a few different little studios and they were all using openPipeline.

B: Oh wow, that's cool.

I5: It's kinda crazy. Yeah. Yeah. It was really weird–it was just weird. Like they don't ever write you and say hey we're using this and it's great. It's just people just use it and it works and I guess that's why I never heard from them is that they didn't have problems. [laughs]

B: [laughs] Right. Well, in that case, that's a good thing I guess.

I5: Yeah. That's kind of the way it always goes.

B: Yeah. Um, I guess one more thing, and I don't know–this is probably for my own vanity–um, did you have a chance to kind of poke around with the pipeline system we have at the lab, here, while you were...?

I5: Not hands on, but I was definitely impressed um, by it for sure. Had you done the summer class–like had you worked on that in the past or...?

B: Yeah. I–I took the summer class uh, in 2011–um, last summer.

I5: Oh, okay–gotcha. Yeah, no definitely I kind of just watched over the shoulder and I did kind of have a few of the guys kinda walk me through it. At one point to kind of like show me the code. And yeah, it was nice. I mean, it definitely, you know I guess that was introduced by Pipeline Mentor a few years ago here?

B: Yes. Some of that code uh some of it is his and some of it is things we wrote that kind of work with it. We have a structure that's similar to that, and we incorporated a few things from the Studio Delta pipeline that we built last summer, um, and kind of tried to make it the best of both worlds from those things.

I5: Yeah, definitely. It looked good. I mean, it definitely–and it behaved nicely. Ant it was very reminiscent of the way we do things here. Um, for sure. So yeah, and um, it definitely–it was cool. You know–it's the kind of thing where, you know, it would be nice to have a UI like inside of Maya. [laughs]

B: [laughs] Yeah, that's one thing I will pass on to whoever comes after me.

I5: Yes, that's exactly what I was going to say–pass on that note. Um, but yeah, I think that's–with that it would be so nice because then you would sort of have this command line tool and you'd have this GUI and if you built like a Python API around the whole thing, then you would just have this real Python pipeline sort of code base that you could really, like, manipulate from anywhere and that could

be really cool. You know like–I'm going to–oh I need to batch process all my shots, or you know, I need to run this–I need to send all the shots to the farm now and you could just write Python little code and tools and stuff around it and um, that would–that would–I dunno if there is such a thing like that right now, but, um, just sort of like an API around the whole thing I think could really like, take it to the next level.

B: Yeah, that'd be pretty cool.

I5: Just being able to batch process or saying oh–yeah, we need to–we have a review coming up, so we need to, you know, just, whatever, output–output versions of all the models or something like that. Or you know, who knows or I want to try this cloth simulator, so I need to output–like a geocache or something. And you have to run, like run pipeline.rungeocache on this file and it just outputs the cache–you don't even have to open it–like you know, that kind of stuff–stuff starts to be like really powerful and really, really easy. So, but yeah it was cool. Yeah, definitely I was like really impressed, for sure, that there was a pipeline at all which is great.

B: Yeah, well, it was, um, definitely a nice project that my boss put us on last summer after the summer course was over and we, um, kind of tried to give something that students could use, you know, for future summer courses maybe and have the, you know, save them some time so they didn't have to build something from the ground up. Um, and also, we just have it year-round, kind of available for students to uh, do projects and, you know, use that basic structure to kind of organize themselves and I'd really–yeah–I'm glad you mentioned the GUI–I would really like to avoid them having to ever, you know, use the file dialog or have the reference editor or any of that kind of stuff and just have almost like an asset browser type thing.

I5: Yeah. Yeah, I mean hey–integrate it with openPipeline–write it in Python. [laughs]

B: [laughs] I would be happy to contribute to the code base.

I5: I was going to say I'll help you guys out if you go down that road. We could just build something that works for both sort of underlying–underlying methodologies. That could be pretty interesting. In your free time.

B: Right–of course. All of the free time that I have. [laughs]

I5: Exactly. So um, when are you done with school?

B: I'm, you know, working pretty hard to finish all this thesis stuff up so that I can defend sometime next month, and then, um, hopefully graduate in December. And job stuff, hopefully is–is looking up, I think I've got some good–a good shot to go for a job at Studio Delta or I saw that there are actually some TD positions open at Studio Alpha.

I5: Yeah, I was going to say definitely, you know, keep in touch with Interview 1. I think he's–I think he's on vacation right now, but as you're sort of going down that road, um, you know, by all means, like definitely let us know how school is going and all of that. And if you can't get in touch with Interview 1, just give me a call–or get in touch and say um, I think I'm–I think I'm gonna be done. I defended and uh, I think they liked it, and so, yeah and for sure and I think... yeah, I think you should definitely keep and eye–especially since you've had the Studio Delta experience already, like uh–there's a lot–there's an amazing pipeline crew here, um, in every respect, um, like in every department. So I think, um, it should be a good learning experience if that was something you're interested in.

B: Oh, yeah, yeah. I'm interested in being there [laughs].

I5: Alright, well that's cool. But yeah, feel free to get in touch if you have any other questions or, yeah, if there's anything you want me to read or give you feedback on.

B: Awesome. Thanks a lot, Interview 5. I really appreciate it.

I5: Sure, no problem, yeah. Have a great day and I'll talk to you soon.

B: You, too. Talk to you later.

I5: Okay, bye.

INTERVIEW 6

BRANDON: Hey Interview 6 this is Brandon.

INTERVIEW 6: Hey.

B: How's it going?

I6: Okay.

B: Good, good. Uh, finally I got to, uh, catch up with you. Glad we got the chance to speak. So did you have a chance to look over the preliminary question list that I sent you?

I6: Uh, yeah. The one you had sent a while back?

B: Yeah, yeah.

I6: Mmhmm.

B: Okay, cool. Um, well then I guess you know what we're in for, so we'll go ahead and get started. Um, just go over real quick sort of your background, uh, your position there at Game Studio One, what you do, how you go there, um, that kind of stuff.

I6: Ok, um, well I started as a, um, software engineer at Animation Studio One, no, I should go back even further. Um, undergraduate at University One, graduate student at University Two at the, uh, graphics lab there. Um, and then I got a job at Animation Studio One as a 2D/3D software engineer. Um, and that was for Adventure Film One and then I worked on Adventure Film Two as a technical director at uh, mostly just a title change–I mean my actual work didn't change much. Um, so Animation Studio One wound down a bit, so I went over to Studio Delta as a software engineer, um, and I was there for eleven years. And um, some of my later assignments, I was working closer and closer with production...and so my last gig I actually moved over to become a technical director on Adventure Film Three,

working in the lighting department there.

B: Yeah.

I6: Um, I'd worked variously on the 2D films, um, and also, um, I worked on [Studio Delta's old], um, pipeline for doing CG rendering on their 2D films. And then I worked with, um, with look, dev and modeling on the 3D films with tools for that and then eventually sort of worked my way back into lighting. Um, and then after Adventure Film Three, um, I went to Game Studio One as a technole–technical director and I've been here at Game Studio One for, um, a year and a half? A little over a year and a half, um, working on tools for the lighting department.

B: Ok, cool. Um, so wh–does your role at Game Studio One differ significantly from what you did at Studio Delta as a technical director? Like does that job description change a lot between those two places?

I6: Um, well the job title can mean a lot of different things at a lot of different places. Actually it's pretty similar between Studio Delta and Game Studio One. Um, it's just a matter of emphasis, so...um, at Game Studio One, there's a little bit more– It tends to be more support work. I think that's just because they had more shorter projects going through for a while. Um, right now we're in a little bit more of ah, a lull as the next project is ramping up, so now it's more tool work. But that's kind of the usual cycle is–you start out doing...scripts and tools and as things start getting busy, you start doing more and more support.

B: Right. Okay, ah, well that kind of gives us an idea of where you're coming from. Ah, so...we'll launch right into it with the big question first–how would you define a pipeline?

I6: [sighs] um, so a pipeline...so, to sort of define something else a little bit– there's the tools that an artist uses to create, um, whatever their work is...um, and then the pipeline is the part of that that, um, takes their data and converts it and

sends it off so that it's usable by other people elsewhere in the facility. Um, it's also the part that bring in what they need to–to get them started so that they can do their work. So the pipeline is sort of the glue between each of the artists' stations, uh, in a way. And it's–and it's a little hard to draw the line because some of those connections and threads sort of percolate all the way up to where they're–like some of the work they do is setting things up so that the pipeline can do its work. Um, but by–mostly the pipeline is all the stuff that's sort of sitting behind what they see when they're doing their stuff.

B: Mmhmm. Yeah, one of the other–[Interview 2] said it's sort of like uh, like a heartbeat. That, you know, it's something that's necessary, but it's not necessarily something that you'd think about. But it's doing a lot of work in the background.

I6: Right I mean–w–yeah. The best kind of pipelines are ones that you don't talk about because you don't need to. Nobody wants to spend time on pipeline.

B: Yeah, yeah, uh, no artists, certainly. [laughs] Not any ones that I know, anyway. Um, on that, what–what kinda things do you think make up a good pipeline?

I6: [laughs] um...so I think, um, one thing is, uh, structure. So...that you sort of know "If I do this, it'll be okay. Things will work." um, that you know that you're gonna get, um. And there's checks and things that can sometimes happen and people and–and indivi–and individual shows can go all over the map–all over the map on something like that. Um, the data you pass through–you do want it to show up in a sort of standard place. You do want it to sort of be in a standard form. It's–It's the more it's understandable, the more it can be automated. Um, on, on the other hand, um...you do want some flexibility, so that if like something odd happens or you're trying to do something unusual, um, you can cu–get that to go through, too–and it's not so locked down that as soon as you try to do something a little bit different–or you–or–or you try to start making changes and improvements, you don't start running

into that your–the fact that you're fighting against the way everything works.

B: Mmhmm.

I6: Um, so, um, one of the ways you can do that is by decoupling. So, um, there's like a monolithic pipe and everything sort of gets carried through in one great big sort of process-munge. Um, decoupling means okay, i'll do this piece. I'll do this piece. And if you change one piece, then hopefully you're not, um, throwing a monkey wrench into anything other than the things that might just connect to that piece. It's usually not quite so clean, but in a way, it's almost better to have dumb processes where if I put this data in–and–and i'll check and make sure that it's good data when I put it in–then things will just pick it up. The thing like way down on the other side will just pick it up and then it's there and all the stuff in-between doesn't really care–I mean, they'll make sure it gets through, but um they won't freak out if it looks a little bit different than usual because it's not real–really their job. Um, so, um, also the less you have to reprocess files, usually the better, um because if you have to do a lot of conversion and extraction and chewing and munching, then anytime you make a change that has a lot of–of knock-on effects.

B: Right–because you'd have to just do that process again every time you made a change to that.

I6: Right. Um, scalability. Um, where if you if you load up ten things, if you load up 100 things, or if you have to do something like a–a few time it might work. You start getting to 1,000, 10,000 then if things start breaking down, then you've got problems. Um, and usually things don't start out scalable. That's one of the things that you have to develop over time–and sometimes you have to pay–you end up bringing in more complexity to get more scalability. Where you get higher level tools and more sophisticated tools, to manage the complicated scenes, um, but you do want to try to keep the simple things simple, as much as you can and only pay for

the complexity when you want to use it.

B: Right.

I6: But uh, it's usually like the feature animation studios are much more like assembly lines–where if you say hey I just want to do five shots. Okay, it's gonna take a really long time–it could take a really long time to set things up so you can do just those five shots because of all the things you have to set up to get going. I mean, they're geared toward much larger scope of work and to just sort of pop something in an whip something out–the pipeline might not be quite so suitable for stuff like that, whereas, like a smaller place, working on a smaller scale, they may be a little bit more agile.

B: And how, um, how does that translate into–I don't know a whole lot about game development. How does that kind of thinking translate into when you're working on–on games.

I6: Um, so, the group I work with actually does the um, like the short two or three minute pieces.

B: Oh, okay, like the cinematic stuff?

I6: Yeah, I g–the cinematics. It's actually the cinematics group. Um, they do those movies, that go with the games. So they don't actually get involved in with the game development.Um, there is a smaller unit that we have that does like in-game, so they might get some in game assets and run them through like the game engine to get out the images and then, and then, put that together into these clips that play as you you're wondering through the forest or whatever. You might stumble across one of these events when you're doing a quest or something.

B: Right.

I6: Um, the game teams tend to–I mean–In a lot of ways, they have similar problems–they've got assets that they're trying to build to put into their system that

are available, um, for the game to consume–the same way that you might have an asset that you want to render. Um, there's usually a little bit of different emphasis on things, um, just in terms of the games have–might–have a lot of people putting things in, so they need to have people watching what kind of work is coming in at–at any point. So I might check something in that might sort of mess up the quality of the game–um, usually, I mean, the same thing too might happen in the cinematic, but that will come out in the render. That won't come out in like the middle of someone playing like the third act of some game.

B: Mmhmm.

I6: Um, so, but you do–you do want to have good quality control because if someone puts stuff in that's broken and it just brings you to a halt and you have to go back and like get it fixed and um, so you have to have a way to sort of deal robustly with "hey maybe this data that you gave me–the character's hair is missing or it's off in the wrong place or his suit turned pink" or something like that. Okay, um, what do you do? And some places it's like well we have everything versioned, so let's go back to the last working version and other places you're like well, let's fix it quickly and get–just get it back to something good. And it's sort of a difference in philosophy because you may want to be able to have–like Sony for example has a lot of projects coming through a visual effects house where they've got a paying client and they say hey I like the look of the thing last month. You ought to be able to give them the thing last month and they can bring that up.

B: Yeah.

I6: Other places it's like we're our own clients, we're our own place, we just need it to look like the way they told us to make it look and it doesn't really matter how it looked a month ago because that's not really what they wanted–they wanted–they want whatever look they have now to get adjusted however they want it to get

adjusted, so they just sort of live in the present and you can maybe look back at how things looked in the past, but, they're always kinda moving forward.

B: So then that would mean that those version tracking tools wouldn't necessarily be as robust as a–like a visual effects place where they might need to call up something that they had a month ago and get that exact asset.

I6: Well, they, they may not even exist. So I know, at Studio Delta, um, they've gone to pretty much a versionless system where if you say "hey I want to render with a look from a month ago," people look at you funny like–but that was last month. And they've, they've out and up decided we don't want to work that way. If you ask for a change, you'll get the change.

B: Yeah.

I6: Now–now something–they do have a little bit of a version control system where if something goes out broken they can say okay, let's revert, but everybody gets it. So, and they've also continuously rendering everything that changes, so they always know hey, you asked for this change this is how the change looks–Is this how you wanted it? The answer's yes, we'll move on, if the answer's no, we'll fix it. Um, so, they're always kind of living in the present. Um, and not–they keep track of the past just because you never want to ever lose data, but they don't ever go back to the past, I mean, um, unless he's like hey I want, turns out, three versions ago. Like we've introduced something where like our render times go through the roof, they're like okay, let's go back to the last time we didn't have that problem. Roll that out, fix it and then we'll update it.

B: Mmhmm.

I6: Um, other places, you can pick and choose versions. They may not work together very well. Um, things may act–act up strangely, but–they let you do that. So, I mean, and there's reasons for each philosophy. And–and it's [sigh] if–If you look

at some places, usually they work is a reaction to the problems they had last time. So, if you have a fresh studio and they're having a problem with hey–ar–they–they gave us new assets and they're–they're broken and they're broken and they're broken and we're having trouble being able to do our work because we can't get on to some working assets and stay there long enough to do any work. It's like okay, let's version control–so, we can stay on the one we know works and when they push out something we don't have to take it. We can–we can wait until we know it's good and then we'll go up to it. Well, then the problem is hey–we made this change–we're not seeing the change. Why is that? Oh, we just haven't updated our assets. Well, are you on the latest version? Eh, not sure. We don't know. We don't know what version everyone's on because they can pick anything.

B: Mmhmm.

I6: Like okay, we need to make sure everyone's on the latest version. Okay. So we make sure everyone's on the latest version and it's like why are we versioning? Everyone's on the latest version all the time...well...maybe we can get–but like it's dangerous to get rid of versions. Well let's make sure that when we push stuff out, it's good. So you–you make that process more robust and then you can choose to get rid of versions. Or you may say you know what? It's a–it's important to us as a facility that when a client says I want I want this previous version. I love this previous version on this previous film. I want–I want a Marvel Avenger's feel. Um, you're like okay, we'll give you Marvel Avenger's feel. Just like they had it. Um, this is valuable to us–It's worth money to us. We're going to put an investment in that–so that we can do this, then you have a place that makes a difference choice. As a–as opposed to some place that says we want to work in the present. We always want to be on the latest version. If someone asks for a change, we want to make sure it happens and we don't want to pay the pipeline complexity for maintaining versions that we never

use.

B: Yeah.

I6: We want to put our money elsewhere. And neither answer is wrong. It's a matter of given–given what your–your, uh, studio priorities are, what is important to you in the pipeline? And what can you do now? Because it may be that someone looks at like Sony's pipeline or Studio Delta's pipeline and says, "I want to work that way." and it's like well, we don't have the infrastructure to do that. What can we do now that may not be everything that they do, but works for us here and we'll start working that way. Uh, and and until you have the infrastructure and the support and the foundation, some of the choices aren't even available to you... or they may not work very well.

B: Those kinds of decisions, like the uh, you know, doing the push versus pull or making those architecture choices, um, who–who decides those things, um, is there just like a secret cabal, you know, that kinda sits in a dark room and kinda figures those things out or how–how are those decisions made?

I6: Um, so that goes to sort of the power struggle of some of the technical leadership between the facility and the show. Um, so if you have a facility where the technical leadership is on–Is in the studio side, so your–your–your software group, your tools group, um, if they're the ones sort of dictating this policy, in terms of how the structure is in that stuff, and the show is merely a client of it. Or if they take the tools and they use the tools and they like them. Whether they like them or not, then, um, then the studio dictates that policy. There's other places where the show is all powerful. If they want something, they get it. Um, up unto the point where they've consumed all the resources they have available to them.

B: Right.

I6: Um, if a show's–it may be one show comes by and is like you know what, we

want to work in a way that is completely push. We want everyone to always be on the latest version. If you didn't want them to get that version, you shouldn't have published it. The next show may come by and go oh, that worked horrible. We're going to be mostly pull. So, your scene won't change until you hit the button and it gets updated. And it can go back and forth between a couple of shows depending on who's the technical leadership on the show and the tools group may be throwing up their hands and saying can you please pick one? Or they may just sort of make it like uh, ok, whatever you want to do–our tools support like either flavor–like we're agnostic. We've built in some flexibility because we've lived through these wars long enough. Um, so it sort of depends where the leadership is–technically where those decisions happen.

B: Right.

I6: Ideally, it's somewhere in-between, uh, where there's some coherent studio strategy or or philosophy that's evolving. And it may be that people use a show as a um, a soap box to make their pitch that the studio should pick a new direction. Um, this happened at Studio Delta where there's a couple of projects that came up where they're like you know what? The way we're working's way complicated–you should simplify our pipeline. And–and, so I mean, there's political kerfuffle, of course, with that,

B: Sure.

I6: But eventually, the–the they decided we'll keep the tools. We like our tools, but we'll simplify our process and make–and rip code out of the tools so we stop supporting some of the stuff we used to support. So they actually got rid of capability and made things simpler, um, as part of this change over. Um, other places, it's like you know what? What if–what if this happens, what if that happens, what if this happens, you know, let's build something that can handle any of those. So you sort

of building tools on sort of a more complex case, a what-ifs case. Um, it's debatable sometimes whether or not like that's necessary, but other times it's like yeah you know what, well, you know it might go one way, might go the other. They might decide they want to push. They might decide they want to pull. They might decide to go back versions. They might–someone might ask for something and we don't want to have to say well, that's gone.

B: Right.

I6: We can't do that, so, build it more complicated to handle that stuff, because we don't want to be looking silly. And that's what you build to. And–and so the–that's just how the dynamic works at a studio. And sometimes they're not really playing it though–they're just trying to get stuff done show after show after show and the pipeline they have is the pipeline they've got. Um, and then eventually, it's like you know what–this is really painful. We should–we should–we should do a project or something to get this fixed up. Um, and then you have the series of either successful or failed projects to try to make it better. And then the projects to replace those projects because obviously they did it wrong the first time around.

B: And those–those kinds of projects–I guess the evolution of a pipeline like reacting to a previous show making those kind of changes, uh, that would be something like, I don't know, Short Film. They wrote a new thing for Short Film maybe or for you know, another shorter production, maybe? It would just be for something like that?

I6: Well, one of the examples, like Studio Delta, they were a 2D shop for a while and then they decided to do chicken little and they said you know what? We know our pipeline doesn't have the capability to do a full 3D movie. So, we need a pipeline upgrade. And that's not even an evolution thing or a reaction thing. It's just a plain old hey we don't have the capability to do what we want to do. Um, and

then, then there's some evolutionary changes over the next couple of films and then for Adventure Film Three they were like you know what, we've been using mostly the same pipeline for a few shows now. It's getting a little long in the tooth. We want to try to do this–try to clean up some of the things that haven't been working so well and sort of redo our pipeline. And then sort of out of that, there was sort of a reaction, okay, you sort of built it that way, but maybe if we simplified it, it would work better. We might have sort have overshot a little bit. And so it–It's dialed back to something that still had the sophisticated tools, but just the way everyone worked was a little bit simpler, um, and a little more streamlined.

B: What kind of feedback loops, um, happen in a pipeline? What kind of communications tools are there that–that aid in that kind of process–the feedback and iteration? Talk a little bit about that kind of stuff.

I6: Um, Studio Delta for example–If an artist is having a problem, there's an email list they send stuff out to. And a bunch of support people see that. And if nobody answers them after a certain amount of time, then there's people who will start poking to make sure they're getting it–a response. And that's sort of their front-line support, um, aside from someone walking over and saying hey–I've got this problem. But those are like immediate day-to-day issues. If something comes up and was like hey, this was a bug or hey I'd like these new features, usually those come up, um, either from people talking or someone sending in an issue with some sort of bug tracking system. Or it comes from, say, the leadership looking at things saying you know what, we will–we have these ideas of how we want to work or we've been hearing these things, we've been–this is difficult–or could even be just they scheduled r&d saying hey we've got–we want to put all these plants in our scenes and the software developers come back with guess what we're going to change your pipeline to make this work out because we've scoped out the work and we realize these are

the ten things that need to happen to get this all to sail smoothly.

B: Mmhmm.

I6: So it even might come from the people developing the pipeline themselves saying hey we've seen this–we've seen what's going on in your pipeline, we've got these ideas for how to make your pipeline better. Or, you know, work with this. Um, so that's part of it. Um, at Game Studio One, there's mostly it's um, uh, someone sends in an issue and then it's um, for upgrades and improvements, it's usually the people who are working on it who have the ideas. It's like hey–hey we should change it to be like this. But they also periodically go to each department and sit them down. It's like okay, what are the things you'd like to see happen? And they–they get everyone together and it's like okay, these are the things we'd like worked on and these are the top–top three. So they go through and ask each department sort of, "what do you want?" And the problem with that is they tend to look department centric and you want to also be looking sort of across the whole studio.

B: Right. I guess because I guess each department may not have that global view of what's happening.

I6: And if you ask someone what they want, you're always going to get an answer. Um, and then you have to decide what's important... Because you almost never have enough resources. Um, there's also the problem where people will, people will tell you their solution. And what you really need is for them to tell you their problem. That's kind of a tricky thing to negotiate. It's like hey I want this–well, hold on. I mean sometimes it's like oh sure, i'll let you have that, but other times it's like why do you want that? Because that's kind of a strange request. I'm not understanding why that makes sense to you. And then they explain it and it's like ooooh. And usually it's–It's hard to do x, so I would like you to do y.

B: Right.

I6: And sometimes the answer is well, what if I made it easier to do x? [laughing] or what if I made it so you didn't even have to do x? We can–we can make x just happen for you like it'll just magically happen and you'll never have to worry about x again. Oh, that's fine–that'll be okay. And so–or other times it's like you know, you're right–you–you it's hard to do x, and so I will give you y. And I will give you the thing you asked for. Um, and other times, it's like x is hard, so I want you to do y and someone else wants z and it's like okay wait a second. For the sake of sanity, we need to sort of settle on which way you guys want to work.

B: Mmhmm.

I6: And so that get's them to–there's kind of this triangle between, um, work flow and pipeline and the tools, um where the way they work is influenced by the tools they have.And the tools they have is sort of influenced by–by the pipeline and how it interacts and that influences the way they work. It's like if all you have is a hammer, everything's a nail. It's like well, what if we had a more screwdriver driven pipeline? Okay, but we've got no screws because everything's a nail. Well okay, we need to bring in some screws. Well, okay, now we've got some screws, but we have no screwdriver. Okay, we need–so it's kind of this chicken and egg thing that tends to happen where, especially with a big pipeline change, where you can't get anything done at the very beginning because you don't know what needs to be done–or you don't know how people want to work, and they don't know how to work because they don't know what their tools are.

B: Right.

I6: And sometimes the answer is okay, let's just take a stab at it. Get something done, and then they'll start telling us what we did wrong and then we can get on with our lives. And part of the reason that can get complicated is often you have different people in different parts of that interface. So that the tool people may be software

engineers off in a different building who rarely descend onto the floor to talk to a um, an actual living artist. Now the TDs might be in with the artists hearing about the complaints everyday and try to sort of glue the pipeline around and working to patch things together, but working almost completely in the dark from what's coming on the tool front. And so there's–there's kind of this, um, it–it really helps to get people together on a–if they're even co-located on a production, because of the communications. Um, software development, broadly speaking, on production is almost unlike any other place, because no one else is going to use your software and your customer is just down the hall.

B: Right.

I6: Right. The whole–"hey I got a bug report and it doesn't make any sense"–you can go over and look over their shoulder and say what were you doing? and watch them do whatever it was that was breaking. Which is extremely rare. And they can come to you and say–or you could just overhear people talking and say hey this sort of sucks and you can–whoa, I'm right next to you. I've heard this. I've gotten information that most people wouldn't bother to send in something about but because I keep talking to people, I've got a better understanding, so. Um, taking advantage of these things helps a lot with your pipeline development because you start getting a better feel for what's going on. And–and just where people sit can make a huge difference.

B: Yeah, I talked to, uh, I talked to [Interview 3] and he was talking about the fact that, you know, if somebody sends an email, like, there's kind of a problem because everyone's so close together, it's really easy to just–just yell at the guy you need to and–and get some help, um, or have him come look at things. Um, speak a little bit about the the differences between doing something on the scale of the cinematics for Game Studio One versus like a feature film, like Adventure Film Three.

I6: Mmhmm. So, for–for the Game Studio One cinematics, usually the projects are a couple of minutes. But, they'll do, um, a lot of art design and sort of concepting and just try to come up with the look and the story and the asset–and de–do all the–design assets. And then they'll start doing it, and then they're done. So, in right about the time that a feature animation would start like hitting its economies of scale and figuring out the process and being able to start turning the crank on the assembly line, the cinematic is over. So it's–it's uh–It's kind of difficult to do short projects because you never–you never really get to hit your full stride. Like about the time you've figured it all out, it's over. Um, and–and it can also happen that like all your shots are coming to you with in just a few weeks of each other. And on a feature animation, that's never gonna happen. You're gonna–you're gonna finish over the course of months.

B: Right.

I6: Um, and you're mostly like on a feature animation, you have to really... be doing planning and scheduling and all of that stuff because if someone sort of decided at the last minute that they want to change the way a couple of the characters look, that has a major implication because you've got 70-80 minutes of footage that might be affected by [laughing] someone changing their mind about something, as opposed to um, okay, we'll–we'll just kinda slug the farm and re-render it because it's only two minutes and we can re-render all the shots that have those characters in them. And that really–we can get away with some of that stuff because we're on a smaller scale.

B: Right.

I6: Um, I've–I've occasionally thought that doing a feature animated film wasn't so much of about doing a film–It was a data management problem. And is o–is exercise in logistics. Um, where as you can sort of squash that over a bit more with the shorter

projects, but in the end it's kind of–you do need to be wary about it. Um, even at two or three minutes, you can overwhelm your resources and not get everything you want into the project if you don't sort of plan it out and think it through. You've just got a little bit more give, usually.

B: So, um, being like you said not really hitting your stride, just kind of figuring things out, is that just because of the uh–amount of work, or is the–the time window for the whole project just much shorter...?

I6: Um, be–part of it is because it's less work. Part of it is because your visual development is in a way almost overlapping with your–actually doing the shots. So your exploration of like okay, we have these ideas–how do they look? What are we gonna settle on? By the time you–on a feature animation do a couple of test shots or do some explorations, and they might use–have a sequence of like okay, this is our first sequence we'll get our first look at this stuff–we'll start really gelling it. With this, well, that's the whole cinematic short. It's just a sequence of a feature animation. Um, in terms of size, so by the time you've sort of settled on what you want, it's over. And by the time you've actually seen everything, uh, clearly, it's time to–to send it on to the game.

B: Right.

I6: So–It's kind of–most–most feature animation studios you'll hear people say, "I wish we could do more shorts to experiment with something. I wish we could do more shorts to sort of try things out without committing to this like 18 month long journey." Um, where yeah, you can make some changes along the way. You can innovate. You can say hey, you know what? Animation wants to work this way–we've got the flexibility. We can sort of add this in. But that's–that's–there's a lot of other things that's like well, we're on the course we're on and we can't really change too much because we've already got 20 minutes under our belt. So 25% of the way in,

you're pretty committed on a feature animation project. 25% in on a two or three minute short, meh. I mean, you've got a lot of assets that you've built and you may not have many people to rework everything, but um, you can sort of iterate–and even if it's like–even if you're sort of stuck, well then there's the next project and the next project, so you might actually be doing two or three projects in a year as opposed to like 0.7 projects in a year. So, you've got a chance to evolve things and try things out and see them in their final form and decide you know what, hey–we're gonna–we're gonna change the way we work.

B: Yeah, you mentioned, uh– flexibility, uh, and being able to handle different things. Does–what kind of processes does that involve? Just like being able to plug in different software packages or maybe change the way that, um, shots bring in assets like what–when you say more flexible, what do you mean?

I6: Um, so if you've got a pipeline that's sort of–you hit a button and it goes looking for things in a particular place with a particular name and it's gonna pick these four things and use them to do whatever it does, if you want to add a fifth thing, then you need to go in and tell it hey there's this fifth thing. It's this new type of data. You need to handle it this way, it needs to be carried through this way and you might need to percolate it through your pipeline that way where it need like–a lot of different things need to be aware of this so that it gets fro–gets from one end to another. A more flexible pipeline just says hey, I've got some stuff. Turns out I've got four things. You tell it oh surprise, I've got five things. Ah, okay, you've got five things, whatever.

B: [laughs]

I6: And it just throws them all in a bucket. And it just carries that bucket along and at various points, someone can say hey, what's in your bucket? And it's like here, I've got these five things and it's like oh, I'm only used to four things; I'm

just going to look at the four. And finally you get to the thing that's looking for the fifth thing. It's like hey oh you've got this fifth thing. I'm a new process I'm going to use that new process you've got there. Oh, okay. So in a sense, it's–that's a more flexible pipeline where it's less aware, in a way, of what anything means, but it can still keep track of things that it doesn't really know about. And the things that need to be aware can pick up on it.

B: Mmhmm.

I6: Um, so, at Studio Delta that was like their blackboard system, where, um, to use a more specific example–If you have a character there's–you could add a block of data on a character or an asset or something that says hey, you've got hair or you've got–I've got data here for the hair system. Um, and there's a whole lot of stuff that wouldn't even care that you've got hair because they don't–they don't have anything–they're looking for your animation, they're looking for hey, what shaders do you have... but every once in a while, something would come in and say hey, I care about hair. Do you have hair? Oh, yes. Okay, well let me do my thing.

B: Right.

I6: Um, if you added a new kind of hair or you added yet another piece of data for like say I'm gonna grow leaves. I've got a very special leaf growing tool. And I've added data to it. It would be very easy to have that in have that get carried through. Um, there's other systems where it's like hey I want to know everything about what's going on. I want to be able to keep track of it. That would be harder to do just because you have to tell it about hey I've got this new type of data, it's this new type of thing. This is the way it looks. This is how you store it. This is how you get it back out. Um, and in a way because it's–It's keeping closer track of things, um, it's harder to make changes. But there may be reasons why you want to work that way. There may be certain value where you can say hey, on all my shots, I need

to know more information about them and it might be hard to do that if you don't know what you've got in all your shots. Um, so for example, like a database based system, for managing your shots, um, might need to know more details about the assets it's containing because it can't store them in the database because it doesn't know what–what this thing is.

B: Mmhmm.

I6: It's like how do you even store it if you don't know what it is? Um, a more flexible system like Studio Delta's, may not have database space, but, um, so there's certain things you may not be able to do very easily but in that case they don't care. Just because of the way their pipeline works. And they do use databases. They just don't use databases for shots, or for shot data.

B: Right. Okay, um, that's pretty much the list that I guess I'd sent and a few other things I'd thought about. Um, is there anything that we didn't address specifically that you were kinda thinking about and maybe wanted to fill me in or is that pretty much it?

I6: Um, hmm. The things like, hmm, the pipeline you have in a place, like there's no one right pipeline. You're generally not going to be able to go and it's like okay, if everyone just did it this way, it would work, because usu–usually you're not starting a pipeline from scratch. You're coming into a pre-existing situation and the first thing that happens is okay, how do we make it work with what we have when what you have is based on what you've been doing? So if you come in and it's like hey, everyone's going to work this way and all this–like you either have to be like such a small shop that you can just sort of drop in a solution like that, or it has to be just a so immensely flexible that it's almost useless and you have to customize and configure it a lot yourself.

B: Mmhmm.

I6: Um, so if you have a pipeline that can do anything at all, then everyone will do anything at all and then no one will know how to talk to each other because one person is doing one thing and another is doing another. So in a way, having–It's kind of balancing between having some limitations and some structure, and having flexibility and support. Um, the more pipeline stuff–and the best pipeline is the pipeline you never talk about. A lot of pipeline stuff boils down to hey can you guys name things consistently? Can you build your hierarchies of your models consistently? And then, as soon as you get into that, then animation and rigging say well wait a second, you need to re-arrange things for our purposes. Hey, we need to name things for our purposes. And it's figuring out how you can do–how you can let people be as flexible as they want to be without contaminating the rest of the pipeline with their flexibility.

B: Right.

I6: Like kinda confine do whatever you want in your box, but don't spread it to everybody else. And as much as you can contain the cor–the chaos and as much as you can keep things consistent, the more you can automate things and the more you can script things and the more–the more, more power you can give–and the sm–the more the building blocks that you're using to do your pipeline, the easier it is for people to build complicated new snazzy things as opposed to sort of anytime I want to send something up to the GUI, I have to write these scripts for composing the job and finding the data and assembling it all into this list and then handing it off. It's like hey, just ask the system what the data is and hand it off to the GUI system and tell it to render and it's gone.

B: Right.

I6: Right, so. Hey, 15 minutes of work versus three days. Um, and again it's not the sort of thing that happens overnight, but it's something you evolve towards. And

then someone says that you–all of your ideas are stupid and they've got it better and they throw it out and they star over from scratch and they realize oh, this is more complicated than I realized. The infamous case of discovered work. People always underestimate how complicated something is when they look at like–this is–why is this so complicated? Sometimes the answer is we're just doing it a dumb way. Sometimes we realize well, we needed to do it that way. Maybe now we can change it, but most of the time it's like hey, we'll just make it simple. And then people will say hey, it's not possible. Hey I can't do this thing I need to do. Or hey, the first system data is getting dropped because you forgot all about that. That was one of my favorite examples. Everyone would try to simplify things and then like what about the first system. Oops! I guess that complicates things.

B: Yeah. Okay, well, uh, I think that's about it. Thanks very much for your time. Uh, I appreciate it. Glad we finally caught up with each other.

I6: Yeah, sorry it took so long to sort of wrangle all the okays and stuff.

B: Yeah, it's okay I–I kinda had to jump through some hoops at some other places as well, so I'm uh, glad we got that all sorted out.

I6: Okay.

B: Thanks a lot, Interview 6. Appreciate it.

I6: Yeah.

B: Have a good day.

I6: Bye.

INTERVIEW 7

BRANDON: Hello?

INTERVIEW 7: Hey Brandon.

B: Hey Interview 7.

I7: Yeah. Hey what's up?

B: I was just calling to, uh, to get this interview started. How's everything going?

I7: Good–busy. [laughs]

B: Yeah. I suspected as much. Uh, did you have a time to look at the questions list that I sent earlier?

I7: Yeah I had a look at them.

B: Okay, cool. Um, well let's get rolling I guess. Um, I gue–we'll ask background questions first, so I guess talk about your position there at Game Studio Two, um, how long you've been there and kind of your educational background and experience.

I7: Okay. So um, so my position at Game Studio Two is um, associate CG supervisor. Um, I've been there seven years... and before I was at–I'll tell you how, um–my educational background is just um, art–is just an art degree. Um, kind of sculpture mostly is what it was. I did some 3D stuff in school. Um, I graduated, [in the early 90s] from college so there wasn't that much, you know, it was kind of uh, the industry wasn't as built up as it is–there weren't that many computer art things as there are now. So...there was this one computer art class in my college and I um, you know, was really into it–me and a couple of guys were really into it–and um, eventually I did–you know, I got an art degree. I worked a little bit in advertising and then I went to a–a–school of communication arts. It's like a degree school for, you know, 3D stuff and I did that and then I got a job at Small Game Studio was

my first game industry job in Location One.

B: Yeah. How did you get, uh, from there to Game Studio Two?

I7: So then I–at Small Game Studio for two years I did animation, um, and then I did–I did mostly like cut scenes–I mean, it was like generalist work back then.

B: Right.

I7: Things weren't as specialized. I did everything, but kind of focused on animation and cut scenes. And then, then I went to Company Three. First, at Company Three [division], where they were doing, um, Joe Network who was the guy who invented ethernet. That'll–yeah, and he did this early [game]–I mean it was an early, early, early game and it was really cool, but he made that game and um, they were–back then they were looking at retained mode for Tech One–this was kind of like a research project for, you know, retained mode and how it might work with, you know, um, gaming and so they just needed artwork so I did all kinds of artwork for it. It was there–like we had, um, I had to do a bunch of power–like little power-up things. The game was a space shooter, um, and it actually got published. It wasn't supposed to get published, but it did. It actually turned out pretty good. It got all kinds of awards and stuff. Um, but there were all these like you can fly after you blow something up–all these power-ups might fly out of them and I think you can buy stuff when you get to–I don't know. We had to manage–or I had to manage really, all of those little power-ups.

B: Yeah.

I7: So um, I'm at Company Three, right? So everyone's a really good programmer. So they helped me–and at [division] which is um, the best people there–so they helped me, um, with uh, doing this like database thing how to organize all the artwork um, so I could find all the power-ups and everything and you know, keep them together. And I thought oh, this is really cool and I kind of started down that path

of technical art. And I don't even know if they had it back then–it was like the new thing; things were different back then–they were more simple. Um, so, you know, I wasn't the first technical artist or anything, but it was way back in the day before it was an established thing. So um, then after that game did really well and I got moved into the regular gaming, you know, Company Three games, um, and I worked on, um, we called it Magic Game and it was RPG um, you know, MMO RPG. And uh, you know, I worked on that for like two more years or three more years, and um, it–then they–they in the middle of–right at the end of making Space Game, they, um, had the idea of making Tech Two. And um, so we started working on Space Game–I mean, um, Magic Game and you know, at that same time, the hardware people were developing Tech Two–and a couple of years into it, they reorganized all the games and they wanted to focus on, you know, Tech Two. And they also wanted to focus on developing out-of-house instead of development in-house because they had a lot of in-house development–development going on for PC. They're like okay, we're going to digress from PC and we're going to put our dollars into Tech Two, but we're not going to do it in-house, we're join to do it out-of-house, so they laid-off almost everybody. I was–I was not everybody–they kept Old Game and I think one more thing. They kept Sub-Studio and they kept Old Game and I think everybody else they kicked out, so...

B: That's brutal.

I7: Then I was like okay. That's when I went to Game Studio Two. Um, and um, I've been at Game Studio Two for like seven years. I started with–oh and while I was working on Magic Game, um, I started doing more–you know, I was doing more and more of the tools, you know, pipeline stuff. Pipeline wasn't as much of a big deal, but it was more about tools, um, at that point. Um, I did some stuff for shaders. I forget, um, what this thing was at this point, um, I did some shaders, um, I worked

with some of the people who were developing the level editor, um, that kind of thing. Then when I went to Game Studio Two, I was–I got hired as a senior technical artist and it had become at that point, you know, a real job. Um, so, yeah.

B: Awesome. Um, and you–you said you've been at Game Studio Two for seven years now, right?

I7: Yes, I think maybe even eight–seven or eight. I started as a senior technical artist–started–what they hired me for, um, they were–there were two big things going on. They were ramping up a lot of people for Movie Game and they were, um, they were doing–they were also moving into like handhelds. So I got hired to go into the handhelds group, but it wasn't formed yet. I guess they were just, you know, getting people together. I worked a little bit, you know, starting out on like Racing Game, and I think I did some stuff on Sports Game. Then we got the um, I did tools. I did like a tool for making this special kind of little edge walls that they make in the, you know, Racing Game tracks. They wanted to procedurally make that–you know, it's just like a barrier–a bar, and then little posts. You know, obviously [laughs] instead of making those, somebody should make a tool.

B: [laughs] Right.

I7: Yeah, so I did that. Um, I forget what I did–it had something to do with, um, the–um, um, file pictures for the players. You know, they got millions of them for the players for Sports Game. I did something with that to help get that together–make it easier to edit them or something like that. Um, I moved in to the handheld–you know where we started up our little handheld division, um. And um, it was, I think the first thing was, um, Action Game for Portable Console–and Portable Console had just come out. It was a launch title–I think it might have been a launch title. But you know, Portable Console was just coming out and we were basically porting Action Game from, I don't know–Home Console I guess? Over to um, to Portable

Console. It was mostly like the assets, you know, and being able to rebuild all the, you know, levels and everything into the–no, into the different format for Portable Console. Hold on my son's here–'what's up, buddy? Yeah. Yeah we can do that later buddy, I'm on the phone... after...'

B: [laughs]

I7: So, um, yeah, so that was that. Then they move the group, um, the um, the handheld group–they moved all that up to Location Three. You know, Game Studio Two has Small Division down in Location Two and they have the really big Game Studio Two studio is up in Location Three, and then there's a couple of other ones, but they moved all the development up to Location Three so, um, then I moved on to Different Sports Game. And at that time, they were doing Tech Three or they were getting ready to do the Tech Three and it was the big um, kind of push for–um, you know it was next year, so I got on Different Sports Game and that was like basically taking um, the Different Sports Game engine um, they had done, um the initial like port of the Home Console or you know Tech Two version to Tech Three, so, um they brought it over to Small Division and we kind of redid the engine underneath–you know, all the art assets, which really is a big deal. So...uh, yeah. So we had to remake pretty much everything and redo pretty much everything and you know, it was a great thing because you could basically re-think everything–you know, the entire game–how we produce everything. You get a change to re-think and keep the good and re-do you know, the–pick another step for the things that weren't working out. So, that was a good time. I was on there for like for or five years and then on to Sports Game.

B: Great. Uh, well that's good. I think we got a pretty good idea of your background at this point, so um, we'll move on to the second set of questions. In your experience working at Company Three and Game Studio Two, um, how would

you define a pipeline? And I know that's a pretty broad question.

I7: No, it's good. I would say there's two things–there's a workflow and there's a pipeline. And the pipeline is–the pipeline is the tools, um, you know, be it, [app] or the applications for creating the art, the um, you know code for pushing the art from the source to um, on console or on the destination platform. I think the more important thing is the workflow. And that's the people, how you–you know, the sign-offs, where it goes from–you know where the asset starts like in concept to the customer. I think that's the most important thing. I think people really, um, sometimes focus too much on the tech and not the people because there can be lots of–a lot can go wrong um, handing assets from one group to another not knowing who's in charge, who's owning the asset or knowing who you're–you know, who you're serving when you're making them, so you know, I think, the pipeline–when I hear pipeline, I think of tech. When I hear workflow, I think of really–the real whole process of making the, you know, the assets for a game.

B: Mmhmm. So it's a combination of the communications processes and the technology.

I7: Yeah, the communications and the people and the personalities. I mean, you know, just put–really putting real people in there is important. You know, the documentation, you know, um, how the art leads work with the assets, you know, how the uh, you know technical directors work with the assets, you know, all of that. So yeah.

B: What characterizes a good pipeline? What are some things that–that make a pipeline–that make that whole process effective–or make it better?

I7: Okay, so I think one of the most important things, like in my experience, one of the most important things of making a good pipeline is–really is–again, it's a personality thing. It's having people who are involved in the pipeline speak up and

talk about what's not working–not work around problems, but speak up and insist–and say look there's a problem with the process here–I want it fixed. You know, um, because what I've found, I mean, that's kind of how you get to a good pipeline is having people who speak up and don't accept problems and issues and don't find work arounds and just use those work arounds. If they say okay, I've worked on this for a week and at the end of the week, I want this fixed–that's how you get to a good pipeline. A good pipeline is just um–you know, it's easy to use and it–it doesn't break. You know, it's robust, easy to use, well documented.

B: Um, what are–well you talked a little bit about working on cinematics for games, um, what are the main differences between a games pipeline and a different kind of pipeline–like for cinematics or for–for animation?

I7: Yeah, I think the main difference is just how–you know, how–specific it is. Like if you're doing cinematics, it's, you know, Photoshop, you know .PSDs and you know .TGAs and just doing textures and putting out, um, you know, frames and, you know, TGA frames or whatever. And, you know, compressing them and it's not very specialized, you know, it's all very–very off the shelf. And with um, games, it's extremely specialized. You know, you might use Maya, but you're using special tools written especially for um, you know, what you're doing. Oh–yeah, so you know with the games like um, you know, it's all–and you need it to be really specialized, right? Because you're trying to save memory and load time and all that stuff, so it's all really specialized knowing the games pipeline–it's just harder. It's harder to work with and generally only because–you know you have to do more–you know, like you have to–there might be a need for, you know, some custom format that eventually, um, you're bringing your assets into and that might be, you know, whatever pipeline engineer writes that, um, you know, maybe the tool to compress it or whatever, you know, it's unlikely–that's not really productized–you know what I mean?

B: Yeah.

I7: It's not completely every case gets handled, it's not completely robust, it's not super well documented, more–more–more likely than not, so you know, that's where the um, conflict and the problems can come. How much time do you spend kind of productizing that, um, how much time do you spend documenting it and making it bullet-proof versus the return because the game–we're not in–in the game business, we're not in the middleware business, we're not in the software business, we're in the entertainment business, so you really don't want to spend, you know, all your time making pipelines, um, you want to make–you want to spend your time making um, making um, you know, entertainment.

B: Mmhmm.

I7: You know, I kind of think of–I think of like tools and pipelines–I try to explain to [laughs] like um, you know, with Hollywood, you know, you're making the film. You know, you're making entertainment and the experience.

B: Yeah.

I7: And that's what you need to be focused on. But, you know, it's like the camera is, you know, the lens is–you don't–you don't skimp on that stuff. You use the best camera you can get, you use the best lenses you can get, and that's kind of what the tools and processes are. They're not, you know, Hollywood isn't in the camera making business–they have other people do that, but they're in the editing software and all that, but you don't skimp on that stuff. You really get the best that you need for that job.

B: Right. Yeah, and I guess that when you're making an animation or a film or something, your–you really only have to worry about what it looks like through that camera for that duration of time. Where as in games, you're seeing–you're going to see stuff from every angle for a lot more time, usually.

I7: Every angle for a lot more time and more likely than not, you'll need to use it again and again and again where in Hollywood, you know, it's once and then it's done. And if you make that movie two, you remake the whole thing. In games, um, you export it once and then you export it again in two weeks and you export it again in two more weeks and then it still needs to hold up when you do the next version of the game. If the game's any good, there'll be a sequel, you know. That next year or that next product cycle, you want to use those same assets again, so they really–it's really just a very different, you know, thing where they really need to be, you know, well constructed and well thought out.

B: Yeah. What kinds of feedback loops and communication happens during the production of a game?

I7: Yeah, that's key. Um, that's key. And, you know, it really depends on the team and how well that team is functioning. Um, you know, there's generally this–there's concept and there's a loop inside of concept of getting the concept down right. Um, and then–then it'll go off to be executed–go off to production to get done and I think that's where it can break down. You know, I think the art directors and stuff can be very comfortable in that, um, real creative soft, um, you know loose process of um, concept, but when it gets to execution, um, sometimes it just gets made and then it gets shipped without a lot more oversight. It really depends on the team, but I've seen that that's kind of where the art direction ends. And maybe there's kind of broad art direction on the whole build, but not asset by asset–you know, dailies, that kind of thing. You don't always see that.

B: Yep. And do you–how do you keep people in the know? Is there just–is it email based or do you kind of sit down in meetings with people? What kind of methods of communication do you use?

I7: That can be really hard. Um, email is really–can be really good because you

know, you can–first of all, you can refer back to it however long you want and it's not like–a really good way to get your message across is to go up to someone and to stand there and have a conversation with them. You have instant feedback and that's great, but you lose–there's no documentation of that, so email is good. I like to have, um, ideally I think, you know, everything you need in terms of production pipelines and workflows and stuff is kind of documented in one place. It's like a living document. People are comfortable going in there adding, removing, editing. You know, not like a big monolithic like comb, more of a wiki style thing where people keep it up to date. Um, that's ideal. You can really end up with a lot of tribal knowledge. I think that's kind of how it ends up a lot–where there are experts who have done this process tons of times, you know, and know, you should do it this way. If you have a question about that process, you go talk to that person. Um, again in that sort of way, it just–if you don't do anything that's what you end up with and it works. That's effective except for when the person isn't there [laughs].

B: Yeah [laughs].

I7: Um, so yeah. So, that's what I think is the best kind of compromise is like a loose living document kind of thing. I think–a super good idea that, um, you know, we've–I've done on a couple of projects that is kind of a natural things is like a gotchas page where it's like unexpected–like a wiki page is sort of document–unexpected stuff that happens a lot where you can just go and oh yeah–there was this weird thing and I know, oh yeah, I know–you know it's on that page. And that's real stuff where you can go through the documents and be like oh well I–but like the weird gotchas–they're all in one place.

B: Mmhmm. How important are version control and change tracking methods?

I7: Super–yeah, really super important–and the same thing with deployment, too, of with pipelines and having things hopefully logged so you have good version

notes, you have a release kind of mechanism or um, a way or releasing the pipeline so that you know everyone has the latest pipeline; everybody knows what changes have been made in it, so if there's some weird behavior... oh well yeah, there was code put in around that thing so maybe there's a bug or something we didn't think about. So, um, like what kind do we use or?

B: Yeah, well, talk about maybe some version control methods and some things that work well for keeping track of assets and version and making sure everybody knows what the latest version of an athlete is or something.

I7: Well, we use a lot of–you know, Perforce is something we use all the time, obviously. You're familiar with that, right?

B: Perforce? Um, no I'm not, actually.

I7: Oh, okay. So there's, um, software called Perforce which is more code–it's more used for code, but um, you know, it's a database where you can, you know, have assets in there and it versions everything you can get the latest and that's a good way to do it. At Game Studio Two recently, we've been using, um, databases like um, databases for keeping assets–you know, relational databases so you know hey this asset affects this asset, um, you know, all that business. So we're using relational databases right now. We have internal kind of technology for–that was developed for, you know, keeping track of assets and um, you know it has relationships between the assets so you know if you touch relationships between the assets–if you touch one thing, all the child assets can be taken care of.

B: Yeah.

I7: Um, so that's kind of what we use for assets themselves. We used to use–what was that one? Um–there was an art-centric, um, like version control thing back in the day that we used to use. I forget what it was called. Um, but we kind of got away from that one, so it's perforce or this internal tack that we have. For the

pipeline and stuff, in terms of keeping people up to date, um, we try to have a lot of stand up meetings, um, where you kind of go around and we'll have the TA's and the artists kind of all together in there and production managers and just be like what you're working on, you know, so people will know what's being touched. So that kind of thing.

B: Okay. What kind of flexibility do you need to have in a pipeline to be able to handle different project requirements?

I7: Um, in my experience, I mean I guess if you're using like Game Engine or whatever, you need a whole lot of flexibility. I've only really worked with dedicated, you know, pipelines that are like project based that we don't, you know, have–you know, we have like our sports pipeline with all the tools and workflows around our sports that we can make Sports Game with, we can make Sports Game Two with, but we couldn't make Different Sports Game or anything like that with. [laughs] so we don't have a lot of flexibility, but this is Game Studio Two and Company Three and Small Game Studio, you know, in my experience, they've all been that way–where you can make the one game with the one pipeline and the one set of tools. But, you know, it's great obviously if you have Game Engine or any of those other–we've used Game Platform a little bit on Different Sports Game and that was really great–I mean those are really great pipelines, but there's a give/take, you know?

B: Yeah, cool. Um, so I guess you–because you're dealing with such specific needs for a project, you're not necessarily going to need to say add a new software package in to be able to make stuff, you know, later on?

I7: Yeah, exactly–and we're real feature based, so, you know, we don't need to do, you know, wings. Anything in–actually, we do–we have birds. We have our basic system, we have our–our skeletons. We have x number of skeletons you can use. We have the player skeleton, we have higher res skeletons for our, you know, hero

character and you know, our really high level characters. We have a quadruped for, you know, horses and stuff and I guess we have wings somewhere for that eagle and then uh, you know that's it for if you want another type of character, it would be a whole big engineering thing to get that all–the same thing with everything, so it's all real specific. But we get the specific and it's very tailored to those types of assets, but it's also very efficient, you know. We get exactly what we want, um, and nothing else. So...

B: Right. Who makes the decisions about pipelines? About the kinds of tools that you need? About the things like–even down to naming conventions? You know, who makes those decisions?

I7: That's a good question. Um, I make those decisions, but not–not in a vacuum–by no means in a vacuum. Obviously there are tools engineers who have a lot–you know it's between–it's basically between the run-time engineers and the artists, right? The run-time engineers need XYZ in order to make their game go. The artists, you know, have ABC to work with, so how do you get ABC translated to XYZ? What's the easiest, most direct, simple way to do that? That's kind of my–you know, how I do it. That is my call, that is my job to say alright, the naming convention will be like this and I'll consult with whoever. If we're going to use–I'm not a fan of naming conventions; I avoid them where possible, but you know, if it has to be data in the name, then we'll try to figure out the best way to do that. Um, tools I try to, you know, do the least with tools that I possibly can get away with only because you're adding–there's a cost to tools–they're not free, right? You're giving yourself some added functionality or added benefits, but you know, that's something that you're putting in your backpack that you're going to have to drag around the mountain for the foreseeable future. You've got to make sure that that's something you really want to maintain and keep because you want it to be–it's got to be A+,

you know?

B: Right.

I7: Um, so are you going to be able to maintain that, keep it A+ and it's going to be worth it? All the effort you put into, you know, into just making that tool–keeping it up and keeping it documented and fixing bugs–so many things. So that's a call I make–it's really those–that's why we have CG supervisors–to make those calls.

B: Mmhmm. Now how have–and you don't have to stay specific on Game Studio Two with this questions–but how have the pipeline systems for productions evolved since you first started working?

I7: When I started at Small Game Studio, I don't think I knew there was a pipeline. I'm trying to think back–I mean, this was in the 90s... specifically back to how we got things in the game. But this was not an issue–we didn't have that many assets. Um, you know keeping track of them wasn't an issue, keeping them updated wasn't an issue. I think we used–actually, if I remember, I think we might have used the Company Three [proprietary] format.

B: Oh, okay.

I7: Yeah, because we were making PC–I think we might have used that. It's just–oh–then you gotta export it, okay. I think we worked in Softimage a lot and they've evolved to be real monsters. About to be a problem if they're not–if they're not done really well to the point where we probably like, I don't know how long it takes to build the game. On a single machine to build Sports Game would take days–all the assets on a single, you know, formal build machine. It's just, you know, I don't know how many different asset types we have–maybe 30 or 40 different, you know, asset types. Um, we have three major pipelines–complete major pipelines. All kinds of packages, you know, Tech One and uh, the Nvidia stuff and our own Game Studio Two and millions–tons of our own Game Studio Two packages, so they've–they've

just–they've evolved in complexity in the geometric way that the performance and the fun and playability has involved–you know, the pipelines have evolved that much.

B: Right. Well that's uh, that's the end of the main questions list that I have. Is there maybe anything I didn't cover–anything I didn't ask about that you, um, any thoughts that you have about pipelines that maybe we didn't get to talk about?

I7: No, I think you had a good set of questions.

B: Okay. Well, good. [laughs]. Thanks for taking some time to talk to me–I really appreciate it. I know you're pretty busy.

I7: No problem at all–take care. And hey–email me back or whatever if you have follow up questions or whatever.

B: Okay. Thanks a lot, Interview 7.

I7: Cool, no problem. Good luck.