

Taming the Complexity of Digital Productions

by Jim Callahan
Temerity Software

The complexity inherent in the production of digital content places high demands on production staff in terms of both time and attention which can significantly affect the quality and cost of delivering the final product. The level of sophistication previously associated with large scale projects such as 3D animated films and blockbuster effects sequences are now becoming commonplace in the production of television commercials and games. The depth of production pipelines, numbers of artists required, and sheer data size and complexity of all digital productions continues to increase.

This paper highlights some of the most serious challenges to digital productions arising from this complexity and provides successful strategies for coping with these challenges. A production visualization and automation tool called Pipeline recently created by Temerity Software which implements these strategies is introduced.

About the Author: Jim Callahan is the President and Founder of Temerity Software and has been at the forefront of software advances in the visual effects and real-time industries for more than 10 years. He has designed and implemented ground breaking applications in the fields of fluid dynamics, volume rendering and artificial intelligence in order to solve some of the most difficult problems encountered in large scale productions to date.

The rapid pace of innovation in technology drives a constant renewal of production techniques and methodologies to exploit the potential economic benefits.

The Pace of Innovation

The hardware, software and artistic techniques used in the CG industry are in a state of constant evolution. The pace of innovation appears to be accelerating in recent years. The driving forces behind these changes are both technological and economic in origin.

Continual increases in processor performance and reduction in the cost of memory and storage has made techniques possible which were previously impractical. New versions of software applications are released frequently to take earliest possible advantage of these increased capabilities. Often the hardware and software used by a production changes during the lifetime of a single project. New features available in software applications and plugins encourage rethinking of existing production methodologies and pipelines. Internal development of shaders, plugins and scripts during the course of production can also impact existing approaches.

The creative demands of each production require innovation. No two productions are identical and the unique challenges of a project frequently lead to new solutions requiring changes in production methodologies. This innovation is desirable, since new approaches can lead to dramatic improvements in both efficiency and quality of the final product.

In this environment of constant change, static organizational models and fixed production pipelines cannot be successfully sustained.

Data Overload

Inability to cope with the volume and complexity of data generated during the course of a production can lead to increased costs, delays in delivery and an overall reduction in final quality.

The sheer volume of data generated by a digital production can be staggering and quickly overwhelms traditional organizational methods. The digital assets involved in a production include a wide variety of data types created by a diverse set of software applications. Upgrades to installed software can cause existing assets to become unreadable or introduce unintended side-effects in the interpretation of this existing data. As physical storage resources are depleted, decisions about which assets need to remain online and which can safely be archived and taken offline are often made based upon incomplete and inaccurate information.

During the course of production, many versions of individual assets are created in response to client interaction and internal creative reviews. This process is not always linear. Newer versions of assets may be abandoned in favor of previously created versions as the production proceeds. Assets created at the beginning of a project may later be revisited and revised long after the artists involved in the creation of the asset have moved on to other aspects of the project.

The digital assets of a production are dependent on each other in complex ways. For instance, the creation of a final image frame may involve dozens of render passes, and hundreds of models, textures, animation and simulation data files. These assets are created and edited by many different artists who may not directly interact with each other. When creative changes need to be made to an asset, the complexity of these dependencies can make it difficult to determine how to accomplish the change. The longer assets have been inactive, the harder it is to reconstruct how they were produced and which artists were involved.

As a production grows in size and complexity, each individual's understanding of the structure and state of completion of the project becomes less accurate and more limited in scope. This may lead to a serious reduction in the quality of decision making at all levels of an organization. Mistakes due to lack of accurate information can cause unnecessary or duplicate work to be performed.

Increased costs, delays in delivery and an overall reduction in quality of the project may result from failures to adequately cope with this deluge of data.

Coping Strategies

Successful strategies for coping with the complexity of digital production environments exist which can largely mitigate the negative economic impacts.

The complex and rapidly changing environment of digital production cannot be avoided, but there are successful strategies for mitigating the consequences to production efficiency and for improving the quality of the final product.

All of the following strategies share a common reliance on the concept of metadata or "data about data". The association of highly structured metadata with each digital asset created during the production provides the necessary foundation for tools which can cope with the complexity of the process and the associated problems.

Artist Insulation

Artists should be insulated from unexpected changes to assets by other artists and to the software environment used to edit the asset.

Insulation of assets from changes to the software environment can be provided by associating metadata with each asset which records the specific version of software used to create or edit the asset along with the information about the OS environment required to operate this software. When assets are later edited, the choice of the software used should be determined by the metadata associated with the asset, ignoring any system-wide or user-specific configurations. This tight coupling between the asset and the software used to create the asset insures that the asset can be accurately shared among artists and recovered at a future date, regardless of changes in software environment.

Versions of an asset should be identified by a metadata tag instead of through changes to filenames. In other words, all versions of the same asset should have identical filenames. When filenames are used to identify versions, a change to an asset has a snowball effect for all assets downstream in the production pipeline. Whenever a new version of an asset is saved under a new filename, any references to the previous filename within other assets needs to be updated in order to make proper use of the new version of the asset. These modified assets must then be saved under new names which propagates downstream in an ever increasing cascade of renaming. Manual mistakes in naming may introduce errors which are difficult to diagnose and frequently lead to much wasted effort.

In order for multiple versions of an asset with the same name to co-exist on the file system, they must be stored in different areas. Each artist should be assigned a working area which contains the specific versions of all assets the artist is currently using. Changes made to the assets within one artist's working area then do not affect other artists. In this way, each artist is insulated from unexpected changes introduced by other artists.

When an artist reaches a state of completion when it is desirable to make the changes to an asset in his or her working area visible to other artists, a new version of the asset should be created by copying it into a central read-only repository. Once an asset has been stored in this repository, other artists may retrieve a copy of the asset from this repository into their own working area for use and possible modification. Once an asset is no longer needed it can safely be deleted from an artist's working area since all versions of the asset stored in the repository are permanently preserved.

Visualization

Visualization of the tree-like structure of relationships among assets in a production is a powerful tool in decision making at all levels of the organization.

When metadata is used to record the dependencies between assets, it becomes possible for producers and artists to visualize the structure of production pipelines and status of the production as a whole.

Digital productions tend to form tree-like structures wherein the assets which are the final product of the production are at the root of the tree and the assets which are manually created by artists form the leaves of the tree. Each step along a production pipeline can be represented as a dependency of the asset further downstream on one or more assets upstream in the pipeline. The assets which form the interior or branches of the tree tend to be generated by automated processes such as compositing, rendering and simulations.

When this tree structure is known through metadata associations between assets, visualizations of this structure prove to be useful aids in the decision making process. Visualizations of this tree structure provide artists with a clear understanding of how the assets for which they are responsible integrate into the production as a whole. Metadata allows the easy identification of the artists which are most impacted by changes to these assets. The same information can be used by creative supervisors and production staff to understand the current status of all aspects of the work in progress; it can quickly identify the artists who need to be notified to accomplish any creative change; and the information provided by this process highlights production bottlenecks.

Automation

Regeneration of procedurally created assets can be made more efficient and accurate using asset metadata.

The details of how to regenerate assets created by automated processes should be recorded in the metadata associated with the assets. Using this information, it becomes possible to determine which assets require regeneration and automatically submit the correct set of jobs with the proper execution order to a queue for processing. This insures that no processing power is wasted performing unnecessary work and that all changes made upstream in a production pipeline are accurately reflected in the final product.

Tools of this type also allow and empower artists to easily test the downstream consequences of a change before exposing the changes to the rest of the production. For instance, a texture painter could easily produce a test frame from the final composite even though generation of that image may involve many steps and techniques outside the normal expertise of the artist. Furthermore, this can be accomplished without involving or interrupting any of the artists responsible for initially setting up the series of steps in the pipeline needed to produce the resulting composited image.

Implementation: Temerity Pipeline™

Temerity Software has created a complete production control application called Temerity Pipeline which fully implements these strategies.

Years of experience in digital productions of all scales led Temerity Software to develop a unique integrated production control application called Temerity Pipeline which fully implements these strategies for coping with the data complexity challenges facing modern productions. The tight integration of Temerity Pipeline's asset management, revision control and distributed execution queue overcomes the inherent limitations of ad hoc combinations of existing software from multiple vendors and provides new functionality unavailable in separate components.

Software Insulation: Toolsets

Toolsets define a collection of interoperable software which when closely bound to assets provides the necessary insulation from changing software environments.

One of the biggest sources of errors in a production is a poorly defined relationship between assets and the software which creates or is used to edit the asset. Pipeline introduces named shell environments, called Toolsets, to define a collection of interoperable software applications and plugins. When software applications are launched by Pipeline, they are executed using a toolset instead of any current site or user specific environment. The toolset used to create an asset is stored by Pipeline as part of the asset's metadata. This provides the required insulation of assets from changes to the software environment and insures that different artists will use identical software when accessing the asset. Toolsets also provide an artist with the ability to simultaneously use different versions of the same software application in a seamless manner without any modification of their user environment, simply by launching the application for an asset from Pipeline.

Asset Metadata: Nodes

Pipeline uses nodes to represent the metadata associated with a closely related set of assets.

Temerity Pipeline represents a collection of closely related assets and the metadata associated with these assets as a Node. Assets are registered with Pipeline by the creation of nodes. Nodes may be associated with a single file or with a set of files which make up a file sequence. Examples include image frames from animation or data files generated by particle, cloth or other dynamic simulations. Nodes may even be associated with multiple file sequences of equal length. These additional file sequences are useful for representing multiple image outputs (diffuse, specular, normal, depth, etc.) generated in a single rendering pass.

Each node contains an Editor property which specifies the software used to edit or view the assets associated with the node. This editor is used to launch the software under the toolset environment specified by the node to provide the desired close coupling of asset and software.

Nodes may be linked to represent arbitrary production methodologies.

Nodes may be linked to indicate dependencies between the assets associated with nodes. Collections of nodes linked in this way describe a production pipeline as a tree of nodes. Pipeline provides a graphical tool which renders visualizations of these node trees which may be interactively edited. Using this tool, production pipelines are organically grown during the course of the production, as artists connect nodes they have created to those created by other artists. This allows Pipeline to adapt to the unique characteristics of each aspect of the project and eliminates the need for a production to adopt a fixed methodology.

Nodes may contain actions to regenerate assets which are created procedurally.

Nodes with procedurally generated assets also contain an Action property. Actions are used by Pipeline to regenerate the assets associated with a node using the assets upstream in the production pipeline. For instance, a node associated with an image sequence, may have a MayaRender action which runs the Maya renderer on a scene file associated with one of the upstream nodes after applying the MEL commands stored in a script file associated with another upstream node. Actions are actually Java based plugins which can be extended to support virtually any kind of command line driven software. This makes Pipeline flexible enough to adapt to any production methodology or choice of software.

Artist Insulation: Revision Control

The use of independent working areas and data sharing mediated by a central repository provides the necessary artist insulation.

Pipeline provides the necessary insulation between artists through a node based revision control system. Each artist may create one or more working areas which are modifiable only by the owning artist. Assets are created by the artist within one of these working areas and then registered with Pipeline by creating a node to manage the assets.

Whenever an asset has reached a desired state of completion, the artist may share the changes to the asset with the rest of the production by performing a Check-In operation on the node associated with the asset. This creates a new version of the node in the repository copying all assets associated with the node into a protected read-only area. The artist must provide a short explanatory note which describes the contents of this new version or the changes made since the last version as part of the check-in operation. All node versions which have been checked-in are visible to all artists using Pipeline. However, it remains up to the individual artist to decide when, or even if, new versions of nodes are retrieved from the repository into their working area by performing Check-Out operations.

It is important to note that these check-in/out operations are actually performed on entire trees of nodes rooted at the node being checked-in/out. In other words, a node version actually describes the entire tree of nodes upstream in the production pipeline which contributed to the creation of the node. For instance, when a rendered image sequence is checked-out, the specific 3D scenes, models, animation data, textures and any other assets which were used directly or indirectly in the process of creating the images are also checked-out.

Automation: Integrated Queue

Using node dependencies and regeneration actions, execution queues can be made far more reliable and job submission is vastly simplified.

Pipeline includes an integrated queue capable of managing the execution of jobs on all available processors of the facility including artist workstations and dedicated render farm nodes. Because Pipeline maintains detailed information about how to create all procedurally generated assets through the metadata contained in the associated nodes, the submission of jobs to this queue is extremely simple. An artist merely needs to specify which asset or assets they wish to create or update, and Pipeline automatically determines the set of jobs which need to be submitted to the queue and the order in which these jobs need to be executed.

This approach has several important advantages. A request to generate a single asset, such as a rendered image, may actually require a complex set of intermediary steps involving techniques and procedures unfamiliar to the artist submitting the request. Pipeline insures that all of these steps are performed in the proper sequence and in exactly the manner specified by the various artists who originally constructed the part of the production pipeline involved. Jobs will only be submitted for the parts of this pipeline which are strictly necessary in order to produce an up-to-date result. Conversely, all changes which have been made to any asset upstream in the pipeline are guaranteed to be reflected in the final result.

Visualization: Node and Job Viewers

Powerful visualization tools allow asset metadata to be effectively exploited thereby increasing production efficiency.

Pipeline provides powerful visualization tools which enable an artist to easily inspect the trees of nodes which make up production pipelines and perform revision control and queue related operations on these nodes. The graphical representation of nodes provides clear indicators of the comparison between working area and repository versions of assets and the status of jobs in the queue associated with these assets. In addition, Pipeline includes a diverse collection of tools for inspecting and editing node metadata, comparing individual working area and repository assets, examining the links between nodes and reviewing node histories. Complete control of the execution queue and detailed examination of jobs including detailed execution statistics are provided in a closely integrated manner.

Conclusion

Although the organizational challenges posed by the complexities of digital productions are great, they can now be overcome through the use of flexible tools, such as Temerity Pipeline, which are able to analyze and adapt to this constantly changing environment. By rethinking some of the fundamental approaches to managing complexity, we have been able to create a first of its kind complete production control application.

Pipeline increases productivity and studio profitability by providing seamless control over distributed processing of jobs, revisions and storage management. Studio pipelines are managed through one environment allowing artists to spend more being creative instead of dealing with production issues.

Temerity believes that the development of in-house production solutions and pipeline methodologies can be more costly, difficult to maintain and less efficient than an integrated solution like Pipeline. The tight integration of Pipeline's asset management, revision control and distributed execution queue provides both efficiency and new functionality which cannot be achieved through gluing ad hoc combinations of existing production tools. Pipeline offers a solid yet highly flexible infrastructure suitable for any studio size or type of production.

About Temerity Software...

Temerity is a software and service provider in the visual effects and digital production industry. Our solutions and products are developed to allow a greater focus on the creative process of bringing ideas to life. We develop tools that automate processes, create efficiencies and most of all free up more production resources.

Our goal at Temerity is to lead the industry in production process advancement and to provide our customers with the most concise, flexible, and specialized software services available.

For more information:

www.temerity.us

info@temerity.us