Fortnite: Supercharging CG Animation Pipelines with Game Engine Technology

Brian J. Pohl^{*}
Epic Games - Enterprise
620 Crossroads Blvd
Cary, North Carolina 27518
brian.pohl@epicgames.com

Andrew Harris[†]
Epic Games
620 Crossroads Blvd
Cary, North Carolina 27518
andrew.harris@epicgames.com

Michael Balog[‡]
Epic Games
620 Crossroads Blvd
Cary, North Carolina 27518
michael.balog@epicgames.com

Michael Clausen[§]
Epic Games
620 Crossroads Blvd
Cary, North Carolina 27518
michael.clausen@epicgames.com

Gavin Moran^{||}
Epic Games
620 Crossroads Blvd
Cary, North Carolina 27518
gavin.moran@epicgames.com

Ryan Brucks^{||}
Epic Games
620 Crossroads Blvd
Cary, North Carolina 27518
ryan.brucks@epicgames.com

Figure 1: Fortnite Characters

ABSTRACT

Game engine technology, when applied to traditional linear animation production pipelines, can positively alter the dynamics of animated content creation. With realtime interactivity, the iterative revision process improves, flexibility during scene assembly increases, and rendering overhead is potentially eliminated.

^{*}Media and Entertainment Solutions Engineer

[†]Studio CG Supervisor

[‡]Director - Animation Technology

[§]Sr. Cinematic Designer and Project Director

^{||}Creative Cinematic Director

^{||}Principal Technical Artist

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

DigiPro '17, July 29, 2017, Los Angeles, CA, USA

© 2017 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5102-7/17/07.
<https://doi.org/1145/3105692.3114816>

CCS CONCEPTS

• **General and reference** → **Performance**; • **Computer systems organization** → **Real-time system architecture**; • **Computing methodologies** → **Non-photorealistic rendering**; Motion capture;

KEYWORDS

Unreal Engine, Realtime, Animation, Rendering, Workflow, Pipeline, Alembic

ACM Reference format:

Brian J. Pohl, Andrew Harris, Michael Balog, Michael Clausen, Gavin Moran, and Ryan Brucks. 2017. Fortnite: Supercharging CG Animation Pipelines with Game Engine Technology. In *Proceedings of DigiPro '17, Los Angeles, CA, USA, July 29, 2017*, 4 pages.
<https://doi.org/1145/3105692.3114816>

1 INTRODUCTION

In 2017, Epic Games created a three minute cinematic trailer for the video game Fortnite. This production consisted of six sequences

containing over one hundred and thirty individual shots. By utilizing Unreal Engine to work within a realtime environment, the creative staff could iterate faster and quickly propagate changes to the production team. Artists, now being more mindful of frame rate and engine performance, were naturally inclined to creatively experiment and find simpler and more elegant solutions.

2 PREPRODUCTION WITH UNREAL ENGINE

During the early storyboarding process, even before there was an editor in place, Sequencer was utilized to construct preliminary assembly edits of the trailer. Having a library of assets prepared early in the preproduction process is important to realizing the benefits of realtime production. Access to preconstructed assets makes previsualization closer to that of live action film production where the production designer build sets, the director blocks out action, and the cinematographer shoots the scene.

For the Fortnite trailer, the production team was able to ramp up quickly by drawing on the existing asset library from the Fortnite video game. Many of these models required increasing their polygonal resolution for the final trailer, but nearly all of the game assets were available for immediate use in previsualization.

Along with motion capture performances, these assets were used to build sets and previsualize potential footage. Using Sequencer, the director and rough layout artists could create shots and directly assemble an edit of the whole sequence in Unreal before passing that footage to a dedicated editor. This saved a lot of back and forth time between the rough layout artists and editorial while improving shot continuity and screen direction. Any additional requests for coverage could be provided at the editor's or director's request.

Upon completion in editorial, the cut was re-conformed in Sequencer where the shots were properly broken down, reindexed, prepped for production and then added to Shotgun for production management and shot tracking.

3 SCENE FLEXIBILITY WITH SEQUENCER

Care was taken during the scene setup process in Unreal to organize the project to reflect a traditional show hierarchy consisting of sequences and shots. To accomplish this, a master "persistent" level acts as an overall virtual container of sublevels and level sequences. Sublevels are used to organize scenes and control, via Sequencer, what is visible at different times throughout the trailer. Each sequence has multiple organizational sublevels associated with it. This allows supervisors to divide tasks into disciplinary branches like layout, animation, and lighting/fx while allowing artists to work in parallel. **See Figure 4.**

By using priority overrides within Sequencer, artists can alter the scene by taking advantage of inheritance from the show level to the sequence level, and from the sequence level to the shot. Objects and attributes are set at the top show/sequence level, and then overridden or modified on a shot by shot basis. This allows for the easy setup of lighting and set dressing requirements at the sequence level where it can be shared, or inherited, by all shots in the sequence. Each individual shot can then choose to override any light or object's attributes. This approach of organizing workflow is consistent with the methodology many animation and vfx companies utilize.

4 FACIAL PIPELINE WITH ALEMBIC CACHING

In an effort to bring animated feature level performances to the Fortnite characters, Epic developed a hybridized FBX and Alembic workflow that overcomes the limited number of eight skin influences per vertex found in Unreal's implementation of FBX. Using this approach, artists are no longer limited by skin influences and are able to utilize all of Maya's deformations and rigging techniques.

Characters are animated in Maya and brought into Unreal as two animations; one for the head and one for the body. The head animation is imported as an alembic cache, allowing for a one-to-one match to the to the originally constructed Maya deformations. During the import into Unreal, alembic files are converted into PCA compressed GPU morphs, stored in memory and then blended per frame in real time. Alembic caching was also used to import rigid body destruction simulations. All character body animations are imported as FBX and Unreal's Anim Dynamics capabilities supplied secondary animation details. Facial geometry and rigging statistics include:

- Facial Rig: 200 Joints, Blend shapes, Lattices
- Head Geometry: 150,000 vertices average

Figure 2: Improved Facial Performance with Alembic

5 DYNAMIC LIGHTING AND COLOR CORRECTION

For both the game and the trailer, Fortnite chose to forego baked lighting in favor of the increased flexibility of dynamic lights with cast shadows. This allows shot-by-shot adjustments of lighting values and set dressing placement. To achieve a high quality cinematic feeling, and still run in realtime, the team leveraged existing features like distance field AO to provide soft shadowing on indirect lights, or non-shadow casting lights. For lights that required true soft cast shadows, artists used the newly added Percent Centered Soft Shadow (PCSS) feature on spotlights and directional lights. Final color correction was done via the new filmic tonemapper. Combined with Sequencer, the tonemapper allowed for final shot-by-shot color correction.

6 REAL-TIME EFFECTS AND POST PROCESSING ENHANCEMENTS

Traditional animation pipelines rely on complicated render farms to calculate visual effects and post processing enhancements as separate render passes that must be composited in a dedicated compositing program like After Effects or Nuke. While the use of these kinds of programs have their place, Unreal eliminates a number of these steps by providing those capabilities in engine. Some of the more notable realtime effects used for this trailer include:

- FFT Bloom
- 3D voxel volumetric fog
- Ray marching shaders for rendering clouds and fluid sims
- Render targets to calculate customized volumes
- Connected distant field AO to base materials, enabling indirect light sources (bounce and fill) to not use cast shadows and instead use soft AO shadows.
- Direct illumination (key and any lights that must cast shadows) using real time soft shadows with PCSS

7 REAL TIME FLUID SIMS AND RAY MARCHING SHADERS

The trailer required over twenty five unique husk death effects. Due to Entertainment Software Rating Board (ESRB) restrictions, the FX had to be very non-violent. Each death needed to be lit correctly within the scene and respond to scene forces. By simulating the effect in realtime the results were more changeable.

Volume textures were stored as "pseudo volumes" and laid out as slices in a standard 2D texture. Characters were voxelized using a simple trick known by the production as "shadowmap voxelization" which used four views due to time constraints. This allowed converting character skeletal meshes into "smoke monsters" that could seed the density of the volume textures. Mesh motion vectors were also captured during voxelization and added to the simulation during advection stages. Artists used velocity with a combination of curl noise and normals from the skeletal meshes.

Figure 3: Smoke Monsters

Each sim was controlled via Sequencer. Sims were triggered one frame early to allow capture of motion vectors. Afterwards, the mesh contribution (density and velocity) could be key framed. This

allowed for very rapid iterations vs traditional offline rendered simulations. Pressure iterations were run at half resolution as an optimization. Blueprints were also created that could convert any sim into a flipbook from the specified camera. Flipbooks were used as a fallback plan for any shots that could not afford simulation (for example if too many sims were needed on screen). The 3D simulations were rendered using a standard ray march shader. Extinction color was exposed to give more stylized color controls.

All volumetric storm clouds were rendered using a similar ray march shader, but utilized 3D flowmaps instead of full fluid simulations. An iterative flowmap technique was used to track longer curly whips of clouds more accurately without distortion. Shapes were a mix of procedural layers and then hand sculpted in VR. The velocities used for the flowmaps were also a mix between curl noise and hand sculpted velocities in VR. Procedural shapes were used more for shots showing lots of growth, as artists controlled the growth of the procedural shapes parametrically over the life of a shot.

8 CONCLUSIONS

In traditional 3D animation workflows, the final piece of the puzzle is achieving beautifully rendered results. While this has been demonstrated as obtainable, its usually accomplished through a considerable investment in computer hardware and by following a rigid, linear and non-realtime studio process. This is becoming more and more challenging with the increasing demands on studios to produce high quality content at a reasonable price and within a reduced timeframe. New workflows based on realtime technologies are becoming the next evolutionary step to addressing these concerns.

ACKNOWLEDGMENTS

The authors would like to thank the Fortnite production team for a job well done and Epic management for giving the creative staff the opportunity and support to accomplish this spectacular trailer.

Figure 4: Fortnite Project Content