Character Workflow of Final Fantasy XV

Kazutaka Kurosaka *
Business Division 2, Square Enix Co., Ltd.

Eitaro Iwabuchi †
Advanced Technology Division, Square Enix Co., Ltd.

Figure 1: ***Left:*** *Hair workflow.* ***Center:*** *character final cuts.* ***Right:*** *Character final cut.*

Abstract

When creating the characters for Final Fantasy XV, we were required to create assets that were convincing and seemed like they could exist in the real world. In addition to PBR as a base technology we also used things like Linear Workflow, a skin shader with subsurface scattering effects, and an eye shader with a corneal refractive structure and two-layer reflective parameters. We will now explain the specific workflow we used to create attractive character assets with the underlying support of these technologies.

Keywords: character, work flow, hair creating

Concepts: •**Computing methodologies** → **Computer graphics;**

1 From pre-rendering to real-time

We needed to design an asset budget for Final Fantasy XV that would allow it to run on a game console. The first measure we took was to make a comparison with pre-rendered graphics. We verified how close we could get to the pre-rendered assets (which are essentially unlimited) using real-time assets that are limited both in terms of memory and technology, and then built the technology and workflow that we required on top of that.

2 Development environment

In addition to improving the quality of our assets, improving development efficiency was also an important effort for us. To solve this problem, we used a single DCC tool to finish asset creation and improve development efficiency. To maintain consistency with how things appeared in the Maya Viewport runtime, we implemented a color grading function equal to that of the runtime.

*e-mail:kurosaka@square-enix.com
†e-mail:iwabeita@square-enix.com

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). © 2016 Copyright held by the owner/author(s).
SIGGRAPH '16, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4282-7/16/07
DOI: http://dx.doi.org/10.1145/2897839.2927448

Figure 2: *A comparison of a pre-rendered character asset (left) with a real-time character asset (right)*

3 Hair creation workflow

3.1 Creating hairstyles

Hair is a major factor in determining a character's quality, and is the part that involves the highest production costs. We will use specific examples to introduce our efforts to efficiently create high quality hair.

While styling actual hair might seem to be a roundabout way to approach this, it is a meaningful process for allowing the artist to understand the structure of hair, and greatly aided our later work.

Figure 3: *The actual hairstyle (left) and the in-game hairstyle (right)*

3.2 Conversion to real-time data: Hair reduction

We use Maya's nHair to create hair. Of course, this will not run on the actual console. It was necessary to reduce the file size while

still maintaining nHair ' s high quality. We created polygons that were similar in shape to those in nHair and then baked the nHair results to a texture. Doing this allowed us to reduce memory usage while maintaining the desired appearance. However, this meant that the workload became quite large, resulting in each character ' s hair taking around a month to finish - so we created a tool to simplify the related processes. Using this tool allowed us to reduce the workload by 60

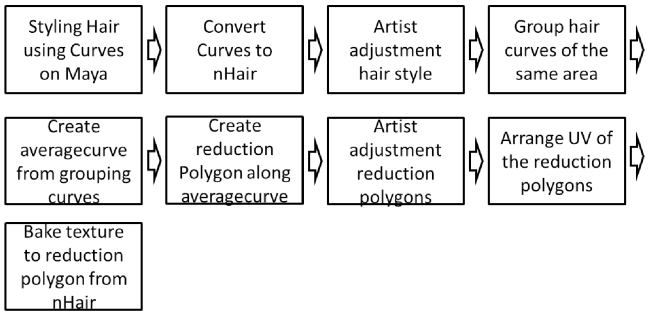

Figure 4: *Hair Reduction pipeline*

4 Character asset post process

4.1 Shaderworks for state changes

The more expressive games become, the more we are required to recreate phenomena that occur in the real world. With this game in particular, the reactions of characters when exposed to magic are one factor that helps to increase immersion in the game, and so this was an important issue for us to tackle.

We will use specific examples using the Shaderworks editor to explain how the characters are used to solve this problem.

Figure 5: *A character state change caused by magic*

4.2 Setting supplementary joints

With normal inverse kinematics animation, violently moving animations will result in the collapse of joint shapes. To solve this, we prepared supplementary bones for each joint and used function control to animate them automatically in order to prevent them from collapsing. (We call this " Kine Driver. ") This solution responds dynamically in the runtime and makes it possible to return to the correct values for additive animations that cannot be recreated using plotted animations.

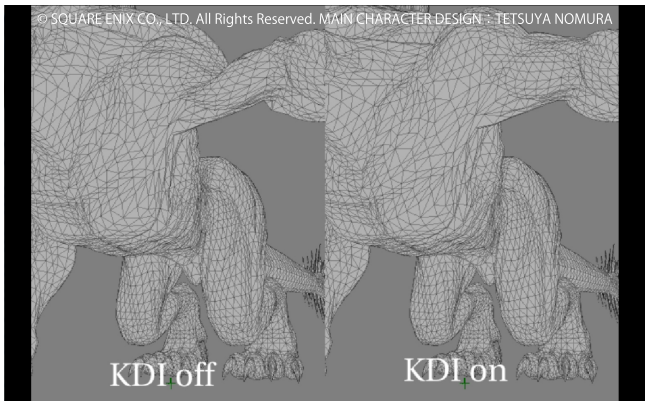

Figure 6: *A comparison showing the supplementary joints turned off (left) and the supplementary joints turned on (right)*

4.3 Cloth-simulation setup

At the preliminary skinning stage, the surface polygons merely follow the motion of the bones. This results in an instant loss of realism. To solve a problem like this, it was necessary to construct a cloth-simulation setup environment. We will show specific examples while explaining what the authoring looked like.

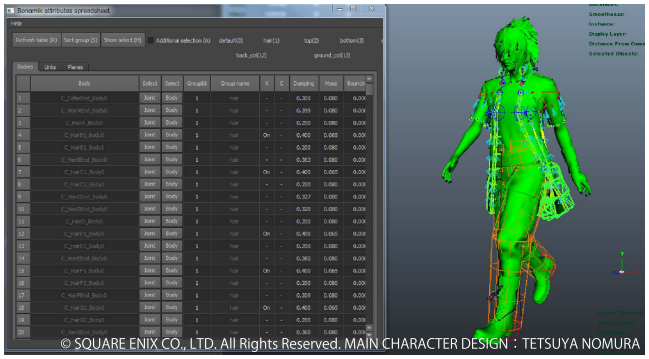

Figure 7: *A comparison of a pre-rendered character asset (left) with a real-time character asset (right)*