Rendering Techniques of *Final Fantasy XV*

Sharif Elcott[1] Kay Chang[1] Masayoshi Miyamoto[2] Napaporn Metaaphanon[1]

[1]Advanced Technology Division, Square Enix Co., Ltd. [2]Second Business Division, Square Enix Co., Ltd.

Figure 1: ***Left:*** *procedural sky and clouds.* ***Center:*** *time-of-day with local probes.* ***Right:*** *result in game.*

Abstract

We present the core graphics technologies used to render and light the world of *Final Fantasy XV*. The latest iteration of the franchise displays a seamless environment more dynamic than ever before, with special features such as dynamic weather, time of day, and procedural sky and clouds. We first describe the hybrid global illumination system based on both dynamic and static elements that is used to handle reflection and bounced light from the different light sources in the game, then move on to explain how the engine procedurally renders sky and clouds and how these dynamic components are intertwined with both the lighting pipeline and the weather system. During the presentation, details on implementation and optimization will be given.

Keywords: global illumination, weather system, procedural sky

Concepts: •**Computing methodologies** → **Rendering;**

1 Game constraints and basic lighting

Final Fantasy XV has an extremely varied environment. The renderer must not only be able to transition seamlessly from indoor locations using a variety of local lights to wide outdoor fields affected by sky lighting, but also be able to handle hybrid situations such as moving train cars with both local lights and wide windows to the outside. As the game makes use of dynamic time-of-day and weather systems, we could not rely on static baked lighting or standard light probe approaches. Since we are working on a wide open-world game, we also had strong requirements on data storage. We therefore decided upon a hybrid global illumination strategy based on both dynamic and static elements in order to strike the best balance for the requirements of the game.

SIGGRAPH '16, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4282-7/16/07
DOI: http://dx.doi.org/10.1145/2897839.2927398

2 Ambient lighting and probe system

2.1 Diffuse light probes

Ambient diffuse lighting is handled with local light probes. Ambient diffuse lighting from the sky is computed using Precomputed Radiance Transfer (PRT) [Sloan et al. 2002]. The PRT transfer matrices are calculated by our in-house path tracer. A layering system is used to blend between the probe grids, which are organized into nested hierarchies. The grids can either be placed manually, as is often done in interior levels, or can be automatically placed to fit the game's navigation meshes or height maps. For ambient diffuse lighting from static local lights that are not affected by time-of-day, we use Irradiance Volumes. Lighting data in Irradiance Volumes is stored alongside PRT information as spherical harmonics. Sun indirect diffuse can be expressed using Light Propagation Volumes (Note: at this point, for performance reasons, LPV has been disabled on console builds) and a custom screen-space ambient occlusion algorithm created in collaboration with the LABS group at Eidos [Michels et al. 2015]. Screen-space reflection is done with a classic ray march at half resolution combined with a roughness-based bilateral blur and upsample [Wronski 2014].

Table 1: *Summary of the lighting components.*

	Local lights	Sun	sky
Specular	GGX	GGX	Pre-filtered cube map / layered reflection
Direct diffuse	Lambert	Lambert	PRT
Indirect diffuse	Irradiance volumes	LPV/SSAO	PRT
Shadows	Dynamic and static shadow maps	CSM / heightmap ray marching	PRT

2.1.1 Moving environments

Among *Final Fantasy XV*'s many varied environments types are those that take place in large vehicles such as trains and airships. When the player is inside such environments they otherwise behave like other indoor environments. The difference is that the environment itself can move and rotate relative to the outside, causing local lighting changes depending on the relative orientation of the windows to the sky. In order to support cases such as these, probes

are optionally baked with a local environment rather than the entire scene, and the SH data is rotated dynamically at run time based on the environment's local orientation.

2.2 Specular light probes

While PRT provides an elegant means of supporting dynamic time-of-day for diffuse lighting, specular lighting is more challenging. The now standard technique of choice for handling specular reflections is a tiered system of screen space reflection, local cube maps with parallax correction, and global cubemaps [Karis 2013]. The problem with this, as it relates to our game's requirements, is that those cube maps must be baked offline and are therefore static. In order to extend this to support dynamic time-of-day, we developed a new technique for storing varying probe data. Unlike techniques which re-evaluate the probes at run time, or which store a mini g-buffer for each cube map and relight them at run time [McAuley 2015], our technique is very fast to evaluate. The memory footprint is also extremely modest compared to fully static cube map approaches. The basis for this method is a novel compression scheme which represents a color-varying cube map (based on time of day, weather, and other factors) by decomposing the final result into a high-frequency component, represented in cube maps, and a time-varying low-frequency component represented in spherical harmonics.

3 Procedural sky and clouds

Final Fantasy XV requires dramatic changes in atmosphere. To give total control to the designers and allow for smooth transitions, we decided to generate the whole sky (including upper atmosphere, sun, stars, moon, and clouds) procedurally.

3.1 Atmospheric scattering

The standard models for simulating atmospheric light scattering in games have many limitations, such as no twilight and lack of intuitive artistic control. In order to overcome these limitations we turned to a precomputed approach. Though this has been done before [Bruneton and Neyret 2008], we found that existing approaches did not match our reference photographs well enough. Our approach uses LUTs generated by a least squares fit to sky and inscatter simulation results computed offline. We use a separate inscatter LUT which allows artists to control the mid-ground color, and a tweaked Mie phase function in order to handle sun occlusion behind distant objects without having to evaluate the occlusion function via a shadow map or other costly methods. In overcast conditions we switch to an entirely different model based on [ISO 2004].

3.2 Cloud rendering

The clouds are modeled using a combination of several octaves of noise functions. The shapes are generated by ray marching. For the direct lighting component we do a ray march between the known heights of the cloud layer and use an extinction transmission map for the occlusion term [Gautron et al. 2011]. For the ambient lighting term we analytically calculate it as an integral over a half-hemisphere of constant color. A compression scheme is used to store the lit HDR clouds in 8-bit textures. Rendering of the procedural sky on screen is done in 3 steps: atmospheric scattering, clouds, and celestial objects and stars. In order to fit within our GPU budget, we amortize the rendering cost across several frames. After the sky box texture is updated, a similar lazy update process is used to perform BRDF filtering for the global specular cube map. Clouds also contribute to the main sun shadow map which lights the remainder of the scene.

Given the slow amortized cloud rendering, care must be taken to avoid artifacts when the clouds are animating. We will describe our cloud animation system, which uses reprojection and temporal blending of in-progress cloud updates in order to give the illusion of smoothly animating cloud shapes.

4 Weather effects

Coupled with the above systems which handle the updates of the sky itself is a robust GPU particle engine for handling rain, snow, fog, and other weather-based effects. In addition, weather changes impact the shading of the rest of the scene, such as causing wetness on surfaces [Lagarde 2012], generating ponds and ripples, and otherwise affecting the lighting balance and post-processing mood.

Acknowledgements

This work is the fruit of the collaboration between the R&D engineers of Square Enix's Advanced Technology Division and the 2nd Business Division. Special thanks to B. Yeoh, C. Haaser, I. Gavrenkov, S. Wilcoxen, C. Ying-I, P. Martishevsky, R. Driancourt and Y. Tokuyoshi for their participation to this work.

References

BRUNETON, E., AND NEYRET, F. 2008. Precomputed atmospheric scattering. *Computer Graphics Forum 27*, 4, 1079–1086.

GAUTRON, P., DELALANDRE, C., AND MARVIE, J.-E. 2011. Extinction transmittance maps. In *SIGGRAPH Asia 2011 Sketches*, ACM, New York, NY, USA, SA '11, 7:1–7:2.

ISO. 2004. Spatial distribution of daylight – CIE standard general sky. Standard, International Organization for Standardization, Geneva, CH, Feb.

KARIS, B. 2013. Real shading in Unreal Engine 4. In *Physically Based Shading in Theory and Practice, ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13.

LAGARDE, S., 2012. Water drop series. https://seblagarde.wordpress.com/2012/12/10/observe-rainy-world/.

MCAULEY, S. 2015. Rendering the world of Far Cry 4. In *Game Developers Conference 2015*.

MICHELS, A. K., SIKACHEV, P., DELMONT, S., DOYON, U., MAHEUX, F., BUCCI, J.-N., AND GALLARDO, D. 2015. Labs R&D: Rendering techniques in Rise of the Tomb Raider. In *ACM SIGGRAPH 2015 Talks*, ACM, New York, NY, USA, SIGGRAPH '15, 81:1–81:1.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 3 (July), 527–536.

WRONSKI, B. 2014. Assassin's Creed 4: Road to next-gen graphics. In *Game Developers Conference 2014*.