WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

MoonRay - DreamWorks Animation Open-Source Renderer Analysis

Anirudh Biswas[1]

Student Animator, Mentor for Computer Graphics
Bachelor of Fine Arts in Animation, Amity University, Noida – 201313
Contact: anirudh.biswas@s.amity.edu

A B S T R A C T

MoonRay is a free and open-source software project developed by DreamWorks Animation that represents a major advancement in animation rendering technology. This ground-breaking framework offers an extensive suite of tools and features, giving artists and developers power to create stunning visual content at unparalleled speed and accuracy. The modular structure of MoonRay allows it to be used as a flexible platform which can be adapted for any animation production pipeline as it grows or changes over time. Collaboration and community-driven development are at the core of MoonRay's design philosophy. By being open about its development process, DreamWorks has managed to create an atmosphere where people feel comfortable sharing ideas around the software leading to more creativity and invention taking place within its orbit. The project continues evolving beyond what was initially thought possible in terms of animated storytelling through continuous updates brought forth by different users who bring with them various skill sets ranging from coding abilities all the way up to rendering expertise thus raising new bars altogether. One area where this system shines brightly lies in how adaptable it can be considered among other applications. Some such features include next-generation light simulation techniques like ray tracing combined with global illumination methods which make each frame appear more real than ever before thus drawing viewers deeper into these intricate worlds around them.

Keywords: *DreamWorks, MoonRay, open-source, rendering technology, ray tracing, animation, innovation, collaboration.*

1. INTRODUCTION

MoonRay is DreamWorks path-tracing animation production renderer, developed in-house and maintained by DreamWorks Animation for all of their feature film production such as How to Train Your Dragon: The Hidden World, The Bad Guys, Puss In Boots: The Last Wish, Kung Fu Panda 4, as well as future titles. Dreamworks engineers built this state of the art - Monte Carlo Ray Tracing (MCRT) renderer. DreamWorks Animation has been awarded a 2019 Entertainment Technology Lumiere Award for the creation of the MoonRay/Arras Lighting Workflow. The production rendering system that can assemble multiple shots simultaneously bringing full production quality scenes to artist desktops in seconds. DreamWorks Animation is recognized by the Advanced Imaging Society and celebrated for its "*distinguished technical achievements in driving the entertainment industry forward with impact through innovation.*"

DreamWorks open-sourced MoonRay on 15th March 2023, under the Apache 2.0 License. DreamWorks continue to demonstrate their commitment to open source projects and their contribution to the computer graphics community. MoonRay is easy-to-use and provides artists with fast iterations It can be integrated into wide variety of tools such as Houdini, Maya, Katana, Blender, in-house lighting tools, etc. with an appropriate plugin or via the hdMoonray Hydra render delegate. It simplifies application integration and also allows MoonRay to take advantage of massive machine scale distributed rendering. DreamWorks expect to see the actual source code to grow even stronger and better with an active community involvement with the openmoonray repository on GitHub. Developers who wish to contribute code to be considered for inclusion in the MoonRay distribution must first complete Contributor License Agreement.

What are some features of MoonRay?

- High-performance path tracing image rendering capabilities
- Stand-alone GUI application for interactive rendering outside of DCCs
- Plugins like HdMoonRay Plugin – supports applications such as Houdini and Maya
- Performs well even without GPU hardware acceleration with pixel-matching XPU mode
- Can be run from a command-line using the RDL scene description language
- Hydra Render Delegate is intended to support batch rendering of USD scenes

- Arras system – used to distribute MoonRay rendering across multiple machines

1.1 Structure

MoonRay was developed from scratch, leveraging state of the art open source components. No studio legacy code was used. The architecture is cleanly divided across three different APIs:

- The Rendering API for clients to initiate high-end path tracing rendering
- The Shading API for the development of pluggable shaders or materials
- The Procedural API for the development of geometry generators

Most of the source code under moonray doesn't require Arras to build, but it is needed by the moonray_arras libraries and by the Hydra plugin HdMoonRay.

- *moonray/scene_rdl2* is the repository that provides MoonRay's scene representation and a number of utility libraries.
- *moonray/moonray* is the repository that implements the MoonRay render engine and the moonray command-line renderer. The code is divided into roughly 20 libraries. moonray also contains a set of basic scene object (shader) plugins.
- *moonray/moonray_dcc_plugins* plugins for using moonray nodes in third-party DCC tools
- *moonray/moonray_gui* implements the moonray_gui Qt application, that can be used to view render progress and final results.
- *moonray/render_profile_viewer* a Python tool for comparing render logs
- *moonray/moonshine* contains a far more extensive set of scene object plugins.
- *moonray/moonshine_usd* has the Usd and UsdInstance scene objects.
- *moonray/moonray_arras* contains the client and server components needed to use MoonRay with Arras.
- *moonray/hydra* has the Hydra plugin for MoonRay, HdMoonRay.
- *moonray/mcrt_denoise* has the denoiser code.

MoonRay uses best-in-class open source libraries.

- Embree acts as MoonRay's ray-intersection engine
- OpenImageIO generically handles different image file formats
- OpenSubdiv is an open source geometry library
- OpenVDB is a volumetric representational format that we open sourced at DreamWorks
- OpenColorIO acts as MoonRay's color management library
- OpenImageDenoise acts as one of MoonRay's denoising tools
- OpenEXR is a deep image format
- USD is a scene description format we use (alongside our own rdl)

1.2 Goals

It was designed to keep all the lanes of all the cores of all the machines busy all the time with meaningful work.

It has a hybrid GPU/CPU rendering mode capable of 100% output efficiency with CPU rendering.

All renderers have personalities, and *"Keep all the lanes busy…"* is MoonRay's personal mantra.

DreamWorks goal was to achieve scalability up to real-time rendering leveraging all of the available hardware. The need to trace and shade billions of rays implied thin interfaces and no data structure redundancy. They embraced "Data Oriented Design", which is a methodology that first grew in the games industry. Apart from DreamWorks' trademark stylized animation, MoonRay is capable of photorealistic output, and has the key features you would expect of a VFX renderer, including AOVs/LPEs, deep output and Cryptomatte. With MoonRay, it ranges between 92 - 154 million core-hours for rendering a DreamWorks Feature.

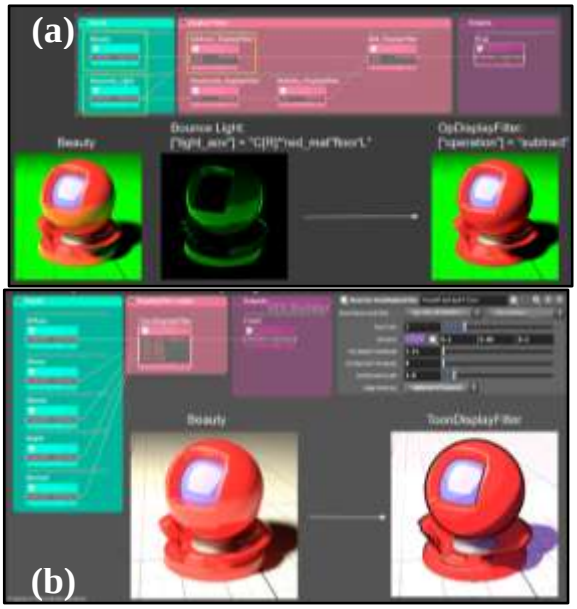
2. UNDERSTANDING THE FEATURES OF MOONRAY

MoonRay is built on a leading-edge, highly scalable architecture with no prior legacy code, allowing quick, feature-film quality artistic iteration using familiar tools. Additional high-performance features include support for distributed rendering, a pixel matching XPU mode that offers improved performance by processing bundles of rays on the GPU as well as the CPU, and bundled path tracing. MoonRay includes a Hydra Render Delegate for integration into content creation tools that support the standard.

Distributed Rendering

This is possible through Arras - a cloud-based framework used for distributing rendering tasks to a cluster or cloud. In the cluster, there is a single MoonRay merge node which is responsible for sending final rendering results back to the client. In addition to enabling distributed rendering, this architecture makes it extremely simple to integrate MoonRay into a wide variety of client applications.

Figure 1: (a) Grading bounce light (b) ToonDisplayFilter



Display Filters

MoonRay provides a set of Display Filters which can serve as compositing nodes that can be piped from AOVs in Moonray along with other Display Filters, which allows progressive refinement in an artist's interactive render session. Useful for non-photoreal and compositing workflows.

XPU Mode

MoonRay's XPU mode is a pixel-matching, production capable GPU accelerated renderer. XPU mode achieves 100% output matching with non-GPU modes and incorporates a graceful fallback-to-CPU mechanism if the GPU has insufficient memory for the scene or other GPU error occurs. XPU treats the GPU like an additional co-processor, submitting work to it while load-balancing the work with the CPU cores.

Hydra Render Delegate

MoonRay comes with a Hydra Render Delegate that is compatible with any DCC tool with Hydra support, for interactive preview rendering. Upon finalization of the Hydra API specification, MoonRay will provide support for final frame batch rendering, and its Hydra Render Delegate will be the supported path to transform USD into MoonRay's internal RDL scene format.

Vectorization

MoonRay was designed to fully leverage Single Instruction/Multiple Data (SIMD) vector units throughout. To achieve high SIMD efficiency, it employs Embree for tracing rays and vectorize the remaining compute-intensive components of the renderer: the integrator, the shading system and shaders, and the texturing engine. Queuing is used to help keep all vector lanes full and improve data coherency.

Profiling Viewer

The standalone Profiling Viewer helps track render performance across a set of canonical tests over time. This tool provides a graphical overview of multiple render logs with flexible options for filtering the results, allowing developers to easily track changes in performance and pinpoint which stages of the render process are impacted.

Adaptive and Uniform Sampling

MoonRay supports uniform and adaptive sampling. Adaptive sampling measures the error in the frame buffer to determine when to stop generating primary samples, taking an efficient non-local view of the frame buffer to eliminate artifacts.

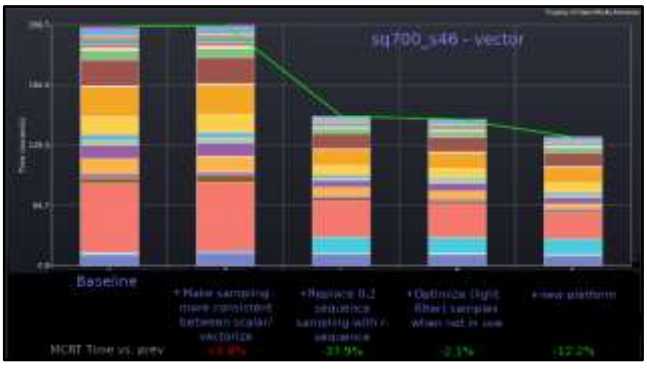


Figure 2: Profile viewer graph chart

Layerable Materials

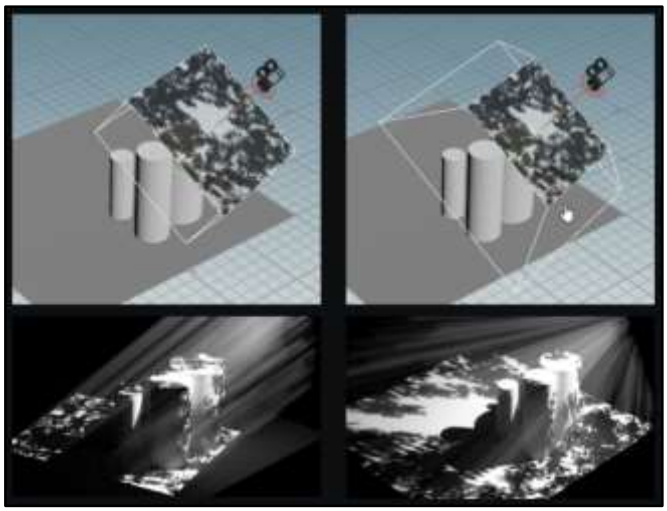
A full suite of DreamWorks' efficient, layerable production materials includes canonical material types for common use cases such as metals, dielectrics, skin, fabric, hair and more. Additionally included is a flexible, general-purpose shader for more complex materials and adapting existing workflows. These materials can be combined using an arbitrary number of layers to represent significantly more complex material types. MoonRay's flexible material shading API allows for easy development of customized material shaders and supports many popular shading models.

Figure 3: Layering Materials example

Light Filters

MoonRay supports a set of advanced light filters for full artistic expression, including Barn Door, Cookie (both isometric and projective), Gobo, Rod, etc. Multiple light filters may be combined together with multiply, min, max, add, or subtract operations.

Figure 4: Light Filters practical example



Volume Rendering

MoonRay supports rendering of participating media such as fog, fire, smoke, clouds, and dust. Features include homogenous and heterogenous media, decoupled ray marching, equiangular sampling, sampled emission, art-directable multiple scattering, overlapping volumes, linear and frustum transforms, motion blur, anisotropic scattering, and OpenVDB file input and baking.

Checkpoint / Resume Rendering

MoonRay supports time, quality and signal based checkpoint rendering, to provide maximal control in many common scenarios.

Deep Images / Cryptomatte

MoonRay supports full deep image output, supporting both the standard .EXR deep format and the coverage-mask based OpenDCX extension. Deep image output is supported for both hard surfaces and volumes in the scene, enabling sophisticated compositing workflows. MoonRay also supports output of Cryptomatte files for generation of ID mattes.

Light Path Expressions / AOVs

All light path expression based AOVs are associated with radiance values. MoonRay supports basic outputs, which describe the differential geometry at the primary ray intersection point, as State AOVs. MoonRay also supports custom outputs such as HeatMap (time per pixel) and wireframe which is used for tessellation diagnostic purposes.

Additionally, Material AOVs provide detailed diagnostics about the materials and are extremely helpful to surfacing and lighting artists when verifying material correctness and standards conformance. MoonRay's Material AOV syntax (intentionally similar to LPE syntax) complements light path expressions, which concern themselves with how a ray travels through the scene. Material AOV syntax focuses on extracting properties of a bsdf at an intersection.

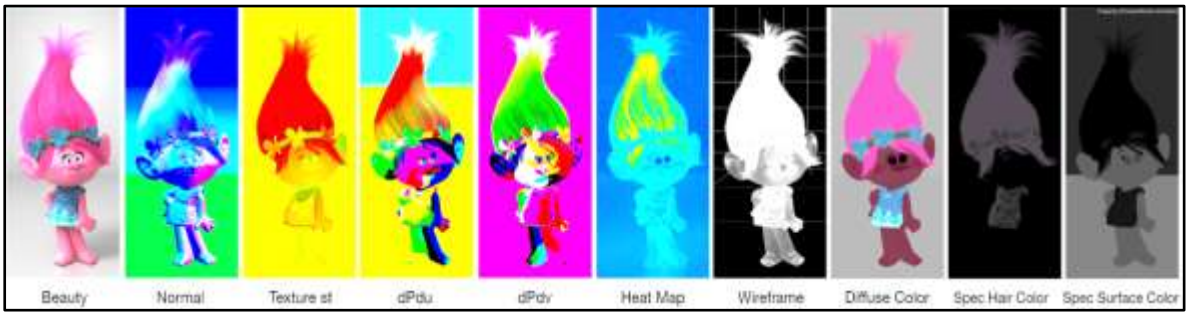


Figure 5: Example AOVs of a 3D character from DreamWorks “Trolls”

2.1 Execution Modes

MoonRay runs in four different execution modes:

1. *Scalar mode*

Scalar mode processes one ray at a time. The rendering is distributed across multiple CPU cores, but MoonRay does not attempt to use the multiple SIMD lanes within the CPU cores for additional parallelism. It also does not batch rays together for improved memory access coherency. Hence, it can be considered a "classical" path tracing algorithm.

2. *Vector mode*

Vector mode achieves higher performance than scalar mode with two strategies:

- Batch rays and shading operations together for improved memory access coherency.
- Process multiple rays and shading calculations in parallel by using the multiple SIMD lanes within the multiple CPU cores.

3. *XPU Mode*

MoonRay's XPU mode uses a NVIDIA CUDA/OptiX-capable GPU to accelerate ray-scene intersection queries. Hence, it is not a complete GPU implementation of MoonRay, but rather uses the GPU as a heterogeneous coprocessor that offloads work from the CPU. XPU mode is designed to pixel-match MoonRay's vector mode output. It utilizes the vector mode infrastructure, hence it inherits the same performance benefits and feature limitations of vector mode. If the mode fails due to insufficient GPU memory, it fallbacks to Vector mode.

XPU mode has the following additional unsupported features:

- Round bezier curves
- Round curves with more than 2 motion samples
- Meshes with more than 2 motion samples

4. *Auto mode*

Auto mode will first try to render in XPU mode. If the scene uses a feature that is unsupported in XPU mode, MoonRay will fall back to vector mode. Then, if the scene uses a feature that is unsupported in vector mode, MoonRay will fall back to scalar mode. This mode will prioritize features over performance.

3. CONCLUSION

MoonRay is capable of high-performance Monte Carlo Ray Traying that keeps all the lanes of CPU cores busy to achieve 100% output matching. It provides with various features that suits the needs of animators and artists to render photorealistic and stylized images. It is important to also note that unlike many other studios, DreamWorks decided to share their proprietary renderer by making it open-source because they believe in their engineering and the community – encouraging and enabling people to explore and build MoonRay further. There was a lot to learn about MoonRay and this research was motivated to bring some awareness on DreamWorks MoonRay technology.

ACKNOWLEDGEMENTS

The author would like to thank and acknowledge the efforts of the DreamWorks Animation Staff for making this into an open-source project and their partnership with Intel, Comcast and other allies. Also, thanks to the developers for their GitHub repository contributions.

References

https://openmoonray.org/

https://docs.openmoonray.org/
https://github.com/dreamworksanimation/openmoonray

https://www.cgchannel.com/2023/03/dreamworks-to-open-source-moonray/
https://youtu.be/MariNCKIXCs [Ubuntu OnAir: MoonRay]