**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 2**

**MPEG TECHNICAL REQUIREMENTS**

**ISO/IEC JTC 1/SC 29/WG 2 m56056**

**Online – January 2021**

**Title: m56222 Annex B Report of characterizing rendering methods for immersive applications**

**Source: WG2 MPEG Requirements**

# Abstract

Rendering forms an integral part of the immersive visual pipeline for presenting content to legacy, current, and emerging displays. As these displays do not follow the same rendering pipelines, when characterizing them it is beneficial to outline these methods to consider the media formats that will be used. This document studies and describes rendering methods and techniques at high-level used by immersive applications. In addition, it provides other rendering considerations that may aid or alter the process.

# Introduction

This document defines methods and techniques used today for the understanding of rendering in immersive applications. As rendering is the one of the ultimate or penultimate step in providing content to the display it is important to define rendering methods as the selection of media may also stem from the certain technique employed.

While these rendering methods have largely been based on synthetic content created in digital content creation (DCC) tools, developments in live action capture to create a 3D geometry or volume from captured images allows for this type of content to be dropped into a renderer for relighting or placed into a computer-generated scene.

**Classification of Rendering Methods**

The following classifies the two main rendering methods. As broad classifications, these methods will be used in applications and displays today but also for emerging and future devices. Rendering engines invoke APIs to use the GPU such as OpenGL [1] and Vulkan [2] from the Khronos Group, OptiX [3] from Nvidia, and DirectX [4] by Microsoft.

* **Real-time:** A real-time renderer accomplishes an output at a rate of at least 24 frames per second. It is essential for real-time renderers to provide visual content at this rate for a smooth experience for the user. Interruptions due to rendering is highly undesirable and can lead to discontent during the experience. Game engines are typically used as real-time renderers as they provide features on top of visual output such as audio, physics simulation, artificial intelligence, and extensions through scripts. Examples of real-time renderers include Iray from Nvidia, Brigade by OTOY, Eevee from the Blender Foundation, Unreal Engine from Epic Games, and Unity 3D.
* **Non real-time or offline**: Non real-time rendering usually outputs individual frames to a specified location that can then be used later in a presentation engine or during final composition. This method is usually used for hyper, photo-realistic, or highly detailed/complex scenes which require processing and calculations on the CPU/GPU which is currently not possible to do in fractions of a second. Offline renderers include OctaneRender by OTOY, Arnold by Autodesk, Cycles by the Blender Foundation, RenderMan by Pixar, and V-Ray by the Chaos Group.

### **Families of rendering techniques for immersive applications**

The real-time or non real-time rendering methods may employ the following widely used techniques to accomplish the desired output. This may be photorealism, but many experiences may strive for non-photorealistic visuals for a cartoon, anime, or artistic look. Nevertheless, the same techniques can be used with differences in shaders or lighting calculations used in the process to achieve a desired look.

* **Rasterization:** This technique follows the typical computer graphics pipeline using shaders to determine the final color of individual pixels. The principle behind rasterization is to generate fragments from inputted vertices through conversion to triangles to determine an overlap with pixels for a final image. The approach may differ per renderer but at a high-level, the 3D geometry is projected from camera space to screen space with projections such as perspective or orthographic. Individual pixels or fragments are evaluated to see whether it is occupied by the geometry then shaded leading to the final visual output of that pixel. Renderers may add steps in between such as processing primitives with geometry shaders or tessellation. Along the pipeline, shaders are used to determine the processing of vertices, geometry, and fragments. This pipeline is used with OpenGL and Direct3D APIs.
* **Raytracing**: Raytracing and path tracing are techniques in which rays are traced from a beginning location, such as the camera, to a specified ending condition. At a basic level, the ray tracing algorithm is simple. By following the path of a ray of light through a scene and seeing how it bounces off objects the light, visibility of objects, scattering of light on a surface, indirect light transport, and its own propagation can be determined to achieve a rendered image.

Path tracing is based on Monte Carlo methods of evaluating the rendering equation which enables better approximations through the average of independent, randomly sampled rays. It is unbiased meaning no error or shortcuts are introduced during approximation of the lighting or radiance. This is beneficial for outputs that are looking to be photorealistic or physically based, however large number of samples are necessary to reduce the amount of noise or artifacts in the final output. This allows for a greater realism in the subsurface scattering of light, specular reflections, global illumination, ambient occlusion, motion blur, and depth of field. The bidirectional reflectance distribution function (BRDF), bidirectional scattering distribution function (BSDF), and bidirectional scattering surface reflectance distribution function (BSSRDF) models are all used for calculation of how lighting simulates with different surfaces and volumes. [5]

* **Hybrid**: A renderer may choose to do a combination of rasterization and raytracing to obtain greater detail in effects like specular reflections, ambient occlusion, soft shadows, or global illumination while optimizing the rendering time. An example of this is hybrid rendering with Real-Time Ray Tracing (RTRT) used in Unreal Engine.

### **Other rendering considerations**

The characterizations above largely assume that the rendering is accomplished on a single client. However, a client may not have the same capabilities to render a scene in the same efficiency as a higher end counterpart. This requires other considerations when it comes to rendering.

* **Distributed & network rendering/edge compute:** Rendering nodes on the network may aid clients by providing additional GPU processing and transmitting the output. The rendering data may be divided for each resource to render a single image over multiple computers or each GPU resource is assigned one frame at a time orchestrated by some form of manager.
* **Scene compression**:To facilitate the transmission of a scene, either live captured or synthetic, compression is favorable. Individual assets of the scene and media types are subject to different compression methods which adds complexity to decompression and rendering.
* **Anti-aliasing**: Artifacts introduced during rendering such as jagged edges can be quelled using with super sample anti-aliasing (SSAA), multisample anti-aliasing (MSAA), or temporal anti-aliasing (TAA).
* **Deep Learning**: Deep learning techniques can accelerate the time to provide rendered outputs through denoising or upscaling. Example usages are AI denoising with a recurrent denoising autoencoder and deep learning super sampling (DLSS) on Nvidia graphics cards. Recently, techniques in deep learning, such as convolutional networks for 3D geometry and manifolds have been researched for many applications including mesh reconstruction, segmentation, classification, and 3D representations based on 2D images or video.
* **Hardware**: Rendering engine performance is powered by the graphics hardware available. Currently, the Nvidia RTX 20 series based on the Turing architecture [7] and AMD Radeon RX 6000 series based on the RDNA 2 architecture [8] are the current consumer top-of-the-line GPUs. The RTX 20 series include dedicated ray tracing cores and the Radeon RX has a Ray Accelerator which both enable ray tracing acceleration. Continued development of these architectures and consideration of dedicated ray acceleration hardware will decrease rendering times for ray tracing renderers. GPUs with older architectures will have naturally have poorer performance in renderers based on newer releases.

# Conclusions

Rasterization is currently employed by all GPUs and is the default choice today for rendering interactive, real-time experiences and scenes as the rendering time is extremely quick. Due to the extremely large number of samples required for a path traced output to eliminate noise which increases the time spent rendering, it has not been the choice of real-time renderers. However, due to the parallel nature of tracing rays enabling acceleration and greater consideration of dedicated hardware on the GPU, a real-time method based on rays may not be guaranteed but is moving closer to reality. This is currently addressed with hybrid techniques and shortcuts. In addition, deep learning can aid the rendering process for path tracing by using less samples and removing artifacts and noise or upscaling through neural networks.

Rasterization may be currently employed for 2D raster displays based on pixels, but this may not be the case for displays emitting rays or waves of light outlined in [6]. Ray or path tracing techniques offer a solution that can be adapted for both legacy and current 2D displays as well as emerging volumetric and light field displays not based on rasters.

The selection for a media type for a display may be influenced by the rendering solution used by the immersive application to present the content. The final visual output of media will differ based on the technique used by the application. For example, a 3D scene with textures will have a different look when put through a rasterization pipeline versus a raytracing or hybrid pipeline. In addition, some media formats may be created with a certain render engine in mind. In this case, the media format may be accompanied with rendering parameters to ensure that there is fidelity in the final visual presentation.

# References

[1] <https://www.opengl.org/>

[2] <https://www.khronos.org/vulkan/>

[3] <https://developer.nvidia.com/optix>

[4] <https://docs.microsoft.com/en-us/windows/win32/getting-started-with-directx-graphics>

[5] Pharr, M., Wenzel J., Humphrey G. *Physically Based Rendering: From Theory to Implementation*. 3rd ed. 2018. [http://www.pbr-book.org/3ed-2018/contents.html](http://www.pbr-book.org/3ed-2018/contents.html%20)

[6] Hinds, T. Arianne. [MPEG-I Future] Draft report on framework for characterizing emerging immersive displays and media for immersive applications. M55594, 10/15/2020.

[7] <https://www.nvidia.com/en-us/geforce/turing/>

[8] <https://www.amd.com/en/technologies/rdna-2>