The Dawn of Real-Time Rendering Solutions: An Exploration into the integration of Real-Time Rendering Solutions in the pipeline for production.

DREXEL UNIVERSITY
Antoinette Westphal College of Media Arts & Design

Stephen Abraham
Department of Digital Media

Abstract

Ever since the earliest renderers were developed in the late 1980s, digital production pipelines have invested a lot of resources in speeding up the rendering aspect of the pipeline. Now with the advent of more powerful Graphical Units (GPU) with innate raytracing abilities, real-time rendering solutions have been massively researched and highlighted upon. The dominant explanation for this trend is with the advent of game engines employing a similar rendering philosophy from the earliest days of video game inception. Previous research has shown especially with the revolution of the Unreal Engines of the significant speed increase in real-time rendering and why more and more video games are shifting towards the employment of technology based on the Unreal Engine system of real-time rendering. With the introduction of Blender's own real-time renderer "EEVEE", the trend shows that almost every major production studio having an interest in researching more real-time rendering options due to the cost reduction it has on the pipeline budget. We set up specific scene renders for three specific renderers- "EEVEE", "Cycles" and "Arnold" following which we compare the respective times and aesthetic comparisons based on the inclusion of Sub-Surface Scattering (SSS) and Light Proxies (LP). The results show that the real-time rendering of EEVEE is significantly faster than the renders of Cycles and ARNOLD on a generic scene level. When we introduce more complicated features such as the SSS and LP, it appears that visually EEVEE fails to calculate the subtle pixel depth for the ray calculation, since it is essentially a rapid rasterization algorithm and hence visually looks visibly different. The findings indicate that while real-time rendering shows extreme speeds for render calculations, it still does not fully employ the Monte Carlo ray-tracing algorithm to accurately define photorealism in its renders.

Introduction

Game Engines have previously been working on the fundamental philosophy that the Graphical Processing Units operates on rendering the frames that appear on the screen; this has been the industry standard for a long time As of 2020, pipelines invested in the Animation, Visual Effects, and the freelancing industries from large scale to indie, and freelance practice, have all begun incorporating Real-Time Rendering Solutions into their workflow[6]; be it through the usage of the Unreal Engine in tandem with Pre-Visualization or advancement of more NPR types of real-time rendering to not just create visually striking images but also with high efficiency as well.

The research conducted aims to analyze the efficiency of real-time rendering solutions against pre-rendered solutions in the areas of speed and aesthetics; this research aims to guide fellow production creators to broaden their horizon in understanding the advantages and disadvantages of implementing real-time solutions into their pipeline and will also allow them to calculate the cost reductions as required

Methods

The methodology employed in this research was to use a comparison study of real-time renderers pitted against an industry-standard renderer that does not have a real-time rendering capability. The highly talked EEVEE renderer of the software Blender was the basis of the experiment, as it is a real-time renderer that has been recently touted for its efficiency in its real-time rendering, a technology that borrows its philosophy from the Unreal Engine system. To be pitted against it, the research employed the use of two pre-render engines; one of which is an industry-standard and one which is a more open-source project. These were the Arnold engine from Maya and the Cycles engine from Blender, respectively. Two scenes with heavy detail were rendered in the three render engines; one scene had the usage of the Sub-Surface Scattering Component of the Renderer and the other scene allowed Light Proxies to be Implemented in the Renderer. These two parameters were specifically chosen because of their uniqueness of these properties. A total of three trial runs were done, to gain a significant sample size following which the times for these specific renderings will be recorded and analyzed.

Results

The results show that when comparing specific viewport renderers, they have massive speed advantages, clocked to almost 30% faster than regular path tracing engines which work on dedicated hardware. It is further surprised to see a more significant increase in efficiency resulting in faster results when you notice that the image quality (in terms of pixel clusters)[1] is practically negligible. What is fascinating is that in this specific example, the lighting samples seem to be over the place especially in the viewport rendering examples. This is because the implementation of the program itself is not due to the regular path results of the three renderers show an interesting development- while the EEVEE boasts of faster times, it fails to capture the essence of the subsurface scattering, and in the scene demonstrating Light proxies, fails due to the fact the EEVEE essentially operates as a rasterization engine and thus does not have the Monte Carlo Ray Tracing Algorithm[5] in it, to allow it to perceive depth as Cycles and Arnold since they are both path tracers that are able to work using the Monte Carlo Ray Tracing Algorithm, and thus are able to calculate pixel perfection realistically albeit with a great calculation time.

$$L_{\mathrm{o}}(\mathbf{x}, \omega_{\mathrm{o}}, \lambda, t) = L_{\mathrm{e}}(\mathbf{x}, \omega_{\mathrm{o}}, \lambda, t) + \int_{\Omega} f_{\mathrm{r}}(\mathbf{x}, \omega_{\mathrm{i}}, \omega_{\mathrm{o}}, \lambda, t) L_{\mathrm{i}}(\mathbf{x}, \omega_{\mathrm{i}}, \lambda, t)(\omega_{\mathrm{i}} \cdot \mathbf{n}) \, \mathrm{d}\, \omega_{\mathrm{i}}$$

Fig 1: Rendering Equation (Lu Yan, 2012)

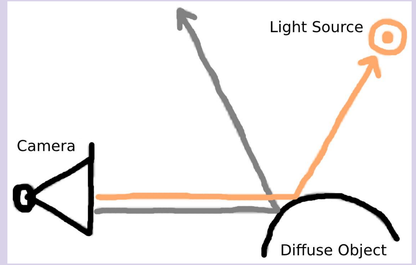

Fig 2: Path Tracing Setup (Greenberg D, 1989)

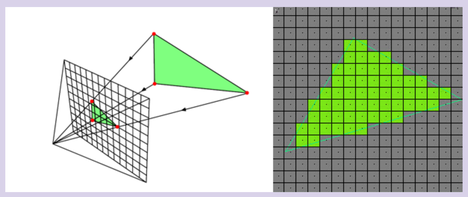

Fig 3: Rasterization (Lu Yan, 2012)

Every pixel's color is determined with shader code based on surface normal and light positions (generally Blinn-Phong & Lambertian shading)[3], edges are anti-aliased, occlusion is determined by using Z-buffer values. Because of this rapid calculation, rasterization engines often must sacrifice greater aesthetic calculations such as the Ambient Occlusion, Z-Depth, and SSS[4]. These however can be usually implemented later and so can be considered as an extra implementation of the pipeline.

Results

Fig 4: Cycles SSS Render

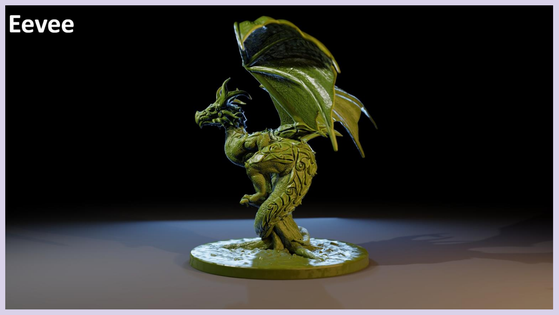

Fig 5: EEVEE SSS Render

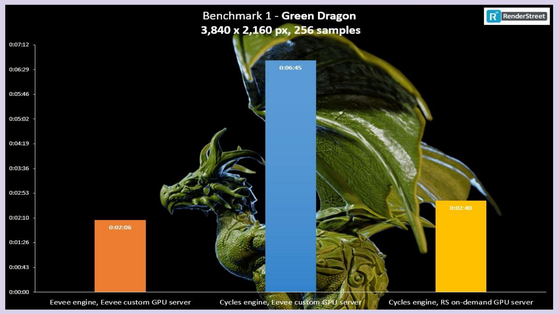

Fig 6: Speed Recordings

Fig 7: Light Proxy Renders

The results show that the objective speed of real time solutions is significantly faster than its counterparts; however, it isn't exactly photorealistic in the pure sense of the term. Rendering in game engines have certainly evolved in a much higher quality renderings in real time renderers such as EEVEE.

Discussion and Future Prospects

The research clearly shows where the issues lie with EEVEE's real-time rendering philosophy; Due to the traditional ray tracing algorithm being abandoned, EEVEE must sacrifice the pixel depth calculation since it must focus on more GI calculations for the faster time. This is essentially what most game engines do. While the standard results do not differ much in scene quality because most 3d objects are often made with opaque surfaces[2]. The fact is that most real-world objects have imperfections and therefore quality of refraction and SSS in their surfaces.[3] Therefore, in terms of hyper-realistic and pure aesthetic purposes, EEVEE fails to counter these issues as suppose Arnold or Cycles would be able to do so. EEVEE has shown that for the first time, within an in-demand software, 3D Artists can use real-time rendering. While it may not be appropriate for use as a final product service, it can be essentially employed as a previsualization tool. The future is certainly bright as there is a heavy push for investment in faster GPUs at a cheaper cost, therefore aiding more production pipelines to use more options of real-time renderers due to the added benefit of a low production cost. It is also admirable to note that while Arnold is trademarked by the Autodesk Company and naturally sells licenses for the renderer, both EEVEE and Cycles are free of cost since they are part of Blender's non-commercial, open-source project. To create a real-time rendering solution that is free of cost and yet maintains the quality of premium real-time solutions such as the Unreal Engine shows the depth and intensity of the research being done at the Blender Studios in Amsterdam. More research into real-time solutions does not just benefit major studio companies but also free-lancers who can contribute and use such a powerful solution with a zero barrier of entry and therefore allowing much smoother and faster pipelines to be constructed and integrated with other various production techniques that could very well contribute to higher volumes of creativity and production content.

Sources

1) Greenberg, D. (1989). Light Reflection Models for Computer Graphics. Science, 244(4901), 166-173. Retrieved August 21, 2020
2) Juran, M. (1998). Simulation Graphics. SAE Transactions, 107, 609-616. Retrieved August 7, 2020
3) Kittler, F., & Ogger, S. (2001). Computer Graphics: A Semi-Technical Introduction. Grey Room, (2), 31-45. Retrieved August 5, 2020
4) Lu, D., & Pan, Y. (2010). Digital preservation for heritages technologies and applications. Heidelberg: Springer. Retrieved August 5, 2020
5) Lu Yan. (2012). The Prospects of Real-time Rendering of Animation Technology. Retrieved August 5, 2020
6) Ayala Dax. (2017). 3D Rendering| What You need to Know. CG Journal,(3). Retrieved August 5, 2020

Acknowledgements

Major credits to-
1) Professor Emil Polyak at Drexel University: Lab Mentor
2) Drexel Applied Research Lab, fellow mentees
3) JSTOR
4) Drexel Summon
5) Google Scholar