Engine Understanding

Callum Wade 2404781 01/10/25

Nanite and the Rendering Pipeline

Introduction

Nanite, Unreal Engine 5's virtualised geometry system, represents a fundamental shift in the rendering pipeline by redefining how high detail 3D assets are processed, optimised and displayed in real time. While both Nanite and teh traditional rendering pipeline aim to take everything in a scene and render it on screen, they achieve this in very different ways, each with it own advantages and limitations. This essay will examine how Nanite fits into the rendering pipeline, compare it with the conventional geometry handling and evaluate the benefits, drawbacks and most suitable applications of each approach.

How It Works - Rendering Pipeline

The rendering pipeline is a critical process that converts 3D models into 2D images on screen, consisting of five stages: Input assembly, Vertex processing, Rasterisation, Pixel processing and Post-processing, all initiated by a draw call. During a draw call, the CPU collects scene data-including vertex/index buffers, shaders, textures and state objects and sends it to the GPU via API calls. Input assembly organises this data into primitives, while vertex processing transforms each vertex from local object space into screen space. Rasterisation converts these vertices into 2D pixels, determining which pixels correspond to each triangle and preparing them for shading. Pixel processing then applies colours, lighting, textures and effects, and post processing adds final visual enhancements such as anti-aliasing, motion blur, depth of field, colour grading and bloom.

Rendering Pipeline Background and Historical Context

The pipeline rendering concept evolved over time, but early examples of the pipeline date back to the 1970s, with the first dedicated hardware for graphics pipelines appearing in the 1980s. According to Paul Joseph (1996) the early stages of the rendering pipeline were developed due to "the power of today’s graphics workstations has increased, so too have the demands of the user. Whereas realism and interaction were previously mutually exclusive, today’s graphic workstations are providing the platform to develop applications with photo-realistic, interactive, dynamic, comprehensible environments." He goes on to talk about how the earliest graphics were primarily just used for early military applications such as project SAGE in the 1950s. But as the "power of today's graphics workstations has increased so too have the demand of the user", the render pipeline was worked on so that the computers can evolve with the demand for better graphical power.

How it works - Nanite

Nanite uses cluster culling within Unreal Engine 5 to render polygons more efficiently then the traditional render pipeline. It does this by breaking models down into tiny clusters and then only rendering the clusters that are visible or needed at the current level of detail. For clusters outside the view or blocked by other geometry, Nanite either reduces their detail or skips rendering them entirely.

Nanite Background and Historical Context

Nanite was released to the public on April 19th 2022 with the goal of artists no longer having to worry about the budget for polycounts, draw calls or memory. When I was researching the development process for Nanite, I read something from one of the graphics engineers at Epic Games. Brian Karis (2021) "How sad is it that once a game is released, the art team posts a ton of their work to ArtStation, and a huge amount of what was posted are high polys and offline rendered images of the models they proudly made but the player never sees look like that?" I feel that this sentence really shed light on why it was important to focus on new and improved rendering methods to allow artists to express themselves further and have more of their hard work actually put into the final project. Not only will it help the artists but it will also mean that companies are spending less money on making assets that won't ever be seen in the end.

Technical Perspective

Improvements from traditional rendering pipelines

Nanite manages to solve the problems that the traditional rendering pipeline couldn't. The main problem was that artists couldn't use too many polygons for their models as it massively increased the file size. But Nanite is able to compress file sizes very well due to its cluster based representation. According to a video I watched (Nanite: Everything You Should Know [Unreal Engine 5], 2021), Nanite is able to compress a one million polygon object down to just 14mb on disc. This is exactly what game developers and artists have been needing to make their games bigger without compromising file size. Another advantage of Nanite is the fact that is only updates things in the frame that change, meaning that it will be much quicker to load each frame, therefore, increasing frames per second.

Limitations With Unsupported Geometry

Despite its advantages, Nanite has several limitations. It performs poorly with aggregate geometry because, while it accelerates rendering, collision is still handled by the CPU, creating a performance bottleneck. Nanite also does not support translucent or masked materials, two-sided faces, deforming objects such as skeletal meshes or tesselation/displacement. Translucency and masking require per-pixel depth sorting, which breaks Nanite's assumption that clusters are either fully visible or fully hidden. Two-sided faces are unsupported because clusters assume a consistent orientation, while deformation is incompatible since Nanite's clusters are pre-processed and static. Likewise, tesselation cannot be used, as dynamically generating new geometry would require clusters to be rebuilt every frame, which is impractical in real time.

Critical Considerations

Performance implications

As well as decreasing the file size of games made on Unreal Engine, Nanite also makes decreases processing time and increases frames per second. This occurs due to clustering objects being a great method for compression and also only objects that have changed in a frame are update, unlike standard rendering pipelines.

Environmental considerations

While being able to make higher detailed assets is a great thing for artists, it can also encourage the creation of extremely detailed scene. While rendering these efficiently on a GPU is optimised, the overall energy demand for game development and hardware usage can increase.

Platform considerations

Currently, Nanite is fully supported on Unreal Engine 5 for PC, next gen consoles (PS5, Xbox Series X&S), and high-end GPUs. However, platforms like mobile devices, VR headsets and older consoles are limited due to a lack of memory bandwidth and computational power required.

Conclusion

The release of Nanite has immensely changed the way that developers and artists work with Unreal Engine and provides more freedom to make bigger and better games without much worry about any form of budget. Overall, even though Nanite does have it's limitations, it is definitely a huge step in the right direction for the future of game development and film creation within Unreal Engine.

References

- Pipeline rendering: Interaction and realism through hardware-based multi-pass rendering - ProQuest (s.d.) At: https://www.proquest.com/openview/c61970ea62b0fbd4c28c517f984d6e62/1?pq-origsite=gscholar&cbl=18750&diss=y (Accessed 01/10/2025).
- Karis_Nanite_SIGGRAPH_Advances_2021_final (s.d.) At: https://advances.realtimerendering.com/s2021/Karis_Nanite_SIGGRAPH_Advances_2021_final.pdf (Accessed 01/10/2025).
- Nanite: Everything You Should Know [Unreal Engine 5] (2021) Directed by William Faucher. At: https://www.youtube.com/watch?v=P65cADzsP8Q (Accessed 01/10/2025).