OPTIMIZATION RESEARCH

Jared Best

Case Study 3: A VR training application for standalone headsets that prepares users for taking certification exams

EXECUTIVE SUMMARY

Task: Focusing on the following case study, research the best practices for creating performant applications as described.

Case Study 3: A VR training application for standalone headsets that prepares users for taking certification exams. For the actual exam, users will be tested on-site with real equipment, so the training application must look and perform realistically as possible. Users must interact with the virtual environment exactly as they would in the real world. Optionally, a second user may log in to the training experience as an observer and must be able to see what the main user is doing as they are doing it, without lag. After performing various tasks, the main user must answer questions about what they did, and these questions must be recorded and sent to a database for later review.

Optimizing VR/AR Experiences

This section discusses general techniques for optimizing VR/AR experiences but all apply to the current case study. The discussion is mainly based on Unity Technologies (2020); however, some information is derived from Unity Technologies (2021a) and Unity Technologies (2021c).

LIGHT BAKING

Light baking is a technique that reduces the computational expense of rendering the Scene by pre-calculating Scene illumination before run time. An advantage of this technique is that there is no run-time overhead for baked lights. Unity renders each object for every light that illuminates it. In other words, if an object is illuminated by three lights, Unity will render the object three times when it is in view. This causes a spike in the tri cound and number of draw calls, degrading application performance. Baking lights is an effective way to reduce the number of tris and draw calls, since these calculations are done prior to run time.

OCCLUSION CULLING

Occlusion Culling is a technique that reduces the workload of the GPU. In essence, it is stopping Unity from rendering objects that are not within the view of the Camera GameObject. Unity by default does not render objects outside of the viewing frustum but still renders objects blocked by objects in the foreground, leading to overdraw or pixel overlap. Pixel overlap can be avoided by using Occlusion Culling to ”cull” any objects covered by other objects.

STATIC BATCHING

Static Batching is a technique that involves combining static (stationary) GameObjects into large Meshes, which Unity can render faster (Unity Technologies, 2021b). This improves both GPU and CPU performance as well as reduces draw calls and physics calculations. If an object is tagged as static, Unity knows this object will not move and therefore does not need to include it in physics calculations. Tagging more objects as static is a helpful technique, especially if the application is CPU-bound.

QUALITY SETTINGS

Quality Settings control the graphical quality of objects being rendered. To access these settings, navigate to **Edit** > **Project Settings** > **Quality**. These are default settings (pixel light count, light map resolution, etc.) that can either maximize graphical quality or maximize performance. The lower the graphical quality, the higher the performance. On certain devices, it might be prudent to lower the graphics quality in order to realize improved performance.

RENDERING PASS TYPE

The rendering method can be adjusted to optimize the VR/AR experience. Unity currently supports three rendering methods for XR: (1) Multi-pass, (2) Single-pass, and (3) Single-pass instanced. The most performant is single-pass instanced; however, it is not available on all devices.

Since Google VR has two Textures, one for each eye, Unity must render a Scene twice. With multi-pass rendering, Unity attempts to avoid duplicating work required for each eye, such as shadows. Multi-pass still renders most objects twice, but renders the Scene graph once. This creates more accurate lighting; however, comes at a computational cost because the two renderings do not share GPU work across Textures. This is the least efficient rendering path but works on most devices.

References

Unity Technologies. (2020). *Unity Learn: Optimizing your VR/AR Experiences*. https://learn.unity.com/tutorial/optimizing-your-vr-ar-experiences

Unity Technologies. (2021a). *Unity Learn: Fixing Performance Problems - 2019.3*. https://learn.unity.com/tutorial/fixing-performance-problems-2019-3?uv=2019.4

Unity Technologies. (2021b). *Unity User Manual 2020.3 (LTS): Draw call batching*. https://docs.unity3d.com/Manual/DrawCallBatching.html

Unity Technologies. (2021c). *Unity User Manual 2020.3 (LTS): Optimizing graphics performance*. https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html

Unity Banner