RESOURCE-EFFICIENT SHADER IMPLEMENTATION FOR IMMERSIVE NARRATIVES ON LEGACY MOBILE VIRTUAL REALITY PLATFORMS

***Uriel Haile Hernandez Belmonte**, Ph.D., professor in the Department of Art, University of Guanajuato, Mexico*
***Fernando Daniel Martínez Cortés**, Digital Art student, University of Guanajuato, Mexico*

Abstract. *In virtual reality applications for legacy mobile hardware, aesthetic decisions are also technical decisions due to limited resources. This research details the resource-efficient shader implementation used for the Sentinella VR game to enable a stylized immersive narrative on platforms like the Oculus Go. The pipeline employs a custom Toon Shader, Inverted Hull outlines, and GPU Instancing to minimize GPU fill rate pressure. This strategy demonstrates that artistic style can be the technical condition that allows a project to run on older mobile VR hardware.*

Keywords: *resource-efficient shader, mobile vr, stylized narrative, oculus go, gpu instancing.*

In virtual reality applications developed for older mobile hardware, the image cannot be understood only as a visual outcome. Every aesthetic decision is also a technical decision. On legacy VR-Mobile platforms such as the Oculus Go, stereoscopic images must be rendered within an integrated system with limited resources, where CPU, GPU, memory, and energy consumption all share a reduced computational budget. For this reason, visual style cannot simply be added at the end of the process as a decorative layer; it must be considered as part of the rendering pipeline from the beginning. The aim of this research is to detail the resource-efficient shader implementation used for the Sentinella VR game and analyze how it enables a stylized immersive narrative to run on legacy mobile virtual reality platforms such as the Oculus Go.

This condition becomes especially evident in the technical specifications of the Oculus Go. The device features a fast-switch LCD display with a resolution of 2560 × 1440 pixels – 1280 × 1440 pixels per eye, selectable refresh rates of 60 and 72 Hz, a Qualcomm Snapdragon 821 Mobile VR platform, and a total of 3 GB of RAM. However, due to system reservations and runtime environments, the memory actually available to the developer is usually reduced to approximately 1-1.5 GB. In this context, the rendering budget is particularly strict, and each visual resource must serve a clear function within the experience [1].

The literature on mobile VR optimization consistently identifies two main pressure points for developers: CPU overhead associated with draw calls, state changes, and command buffer generation in general; and GPU fill rate pressure, especially when full-screen effects or multiple texture samples are involved [2, 3].

To address these restrictions, the rendering pipeline of Sentinella was structured around four main decisions: First, simplifying object shading through a custom Toon Shader instead of standard PBR materials. Second, using an Inverted Hull Pass for outlines instead of a full-screen edge detection technique. Third, applying GPU Instancing to repeated groups of compatible meshes and materials that share the same texture, while using mesh UVs to access different regions of that texture. Fourth,

limiting the post-processing of Sentinella Mode to a single full-screen blit, while using layer masks to isolate special objects and avoid an additional full-frame blit.

The Toon Shader, consistent with its non-photorealistic rendering nature, uses a deliberately simplified lighting model. Instead of relying on a physically based model, the shader calculates a tonal sample from three main elements: the object's base texture, the surface normal, and the direction of the main light. This approach reduces the algorithmic complexity of lighting and material calculation while preserving a stylized visual identity (Figure 1) [4].

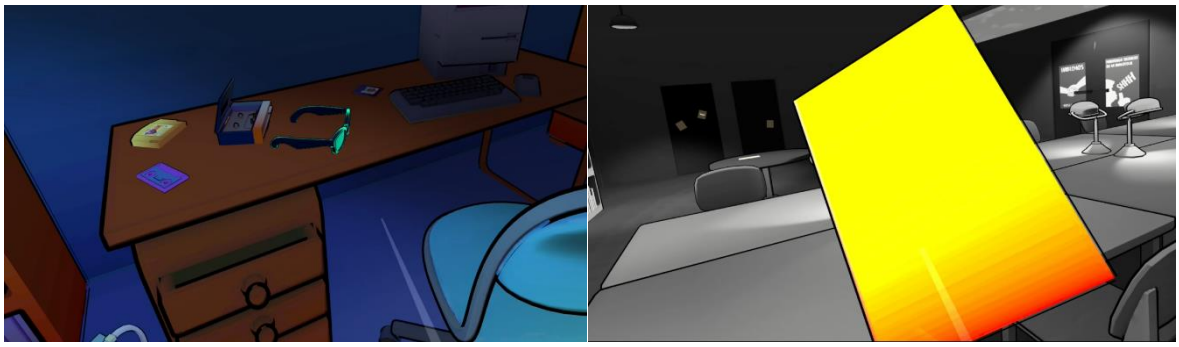

Figure 1 – Comparison between the standard stylized rendering of Sentinella and Sentinella Mode. The left image shows the Toon Shader with outline rendering, while the right image highlights the grayscale post-processing effect and the double-camera system used to preserve selected special objects

In the Inverted Hull pass used to generate the outline, the shader discards front-facing polygons, offsets the vertices along their normals in world space, writes to the depth buffer, and returns a flat color for the expanded silhouette (Figure 1) [4].

This technique is structurally advantageous for Mobile VR because it avoids executing outline processing across the entire screen. In comparison, an outline based on Sobel filtering over depth-normal textures would require one or more full-screen passes per frame, commonly sampling a 3 × 3 pixel neighborhood. For this reason, a Sobel-based method places considerable pressure on fill rate and memory bandwidth, especially on legacy devices such as the Oculus Go. The Inverted Hull approach, by contrast, is not executed over the entire screen. Its cost depends on the amount of geometry using the shader, since it is added as an extra pass to the Toon Shader. However, this introduces a deliberate trade-off between CPU and GPU costs: while a Sobel-based method increases full-screen fragment processing and bandwidth consumption, the Inverted Hull method increases the number of draw calls for the affected geometry.

However, GPU Instancing and batching alter this cost structure. For compatible objects, meshes, and materials can be grouped into instanced batches. In this case, the cost depends on the number of instanced groups rather than on the total number of individual objects. Therefore, the technique represents a deliberate compensation strategy: it increases vertex processing and adds an additional pass, but avoids the cost of full-screen multisample fragment processing, which is especially expensive on limited mobile GPUs.

This approach also has historical support in research on silhouette rendering. Raskar and Cohen [5] describe the use of translated back-facing polygons to create continuous line regions. Although the Inverted Hull technique differs in its specific implementation, it belongs to the same conceptual family: both approaches use back-facing geometry and spatial displacement to construct a visible silhouette. In the case of Sentinella, this displacement is performed along the vertex normals in world space within Unity's shader pipeline.

The post-processing effect used in Sentinella Mode operates on a single source texture. From the UV coordinates, the shader calculates a radial mask and uses that mask to interpolate between the original color and its grayscale conversion [4]. The script receives a source render texture and writes to a destination render texture, resolving the effect through a single blit. The material is cloned only once at runtime, shader property IDs are cached, and the parameters are updated per frame. This is important in Mobile VR, since per-frame allocations and garbage collection can become significant sources of stutter. For special objects, instead of applying an additional full-frame blit, the pipeline uses a temporary secondary camera that is active only while the effect is in use. The main camera excludes the SpecialColor LayerMask and renders the post-processed result. The secondary camera then renders only the geometry assigned to the SpecialColor layer over the processed image. This approach avoids repeatedly rendering unnecessary objects and prevents the creation and rendering of additional intermediate textures. The full shader implementation is available in the project's public GitHub Gist repository [4].

In conclusion, the rendering pipeline in the Sentinella VR game demonstrates how immersive narratives can adapt to the restrictions of legacy mobile VR hardware. Its custom shaders avoid costly calculations associated with standard PBR models while also expressing specific narrative meanings. This establishes a practical balance for devices such as the Oculus Go. The result is a rendering pipeline that treats resource efficiency not as a reduction of the project's artistic ambitions, but as a set of technical conditions and structures that make a stylized narrative possible. In this sense, visual style does not function merely as an aesthetic ornament, but as the technical strategy that allows the project to run on older mobile hardware.

References

1. VRcompare. (n. d.). Oculus go: Full specification. VRcompare. https://vr-compare.com/headset/oculusgo.
2. Android Developers. (2025). Materials and shaders. https://developer.android.com/games/optimize/material.
3. Google VR. (2024). VR performance best practices. Google for Developers. https://developers.google.com/vr/develop/best-practices/perf-best-practices.
4. TaCodeC. (2026). Sentinella_OutlineCelshading_LensCombined.shader [Source code]. GitHub Gist. https://gist.github.com/TaCodeC/2b15c8d403a3071dee795df6dce3ee49.
5. Raskar, R., & Cohen, M. (1999). Image precision silhouette edges. Proceedings of the 1999 Symposium on Interactive 3D Graphics. https://my.eng.utah.edu/~cs5610/handouts/Raskar-Cohen.pdf.