What are the main hurdles to the adoption of foveated rendering in HMDs?

Joe Down

April 11, 2024

1 Background

Humans direct their gaze towards an object by rotating their eyes such that light reflected by the object viewed falls on the eye's fovea[3]. This is a small region at the centre of the retina with increased cone density[5]: the photoreceptor cells of the eye which send signals to brain when stimulated by light[2]. Additionally, the brain is structured to process this information with a space-variant resolution dedicating most resources to the fovea region[9]. These conditions mean that spatial acuity is strongest for objects viewed within this area[3].

In virtual reality systems, it has been found that delay in frame delivery results in motion sickness, discomfort[7], and decreased performance in certain motor tasks[6]. Since rendering requirements continue to increase as screen reolutions and scene complexities increase, it remains difficult for modern computing devices to render complex VR scenes with low visual latency[8]. Foveated rendering has frequently been proposed as a technique to reduce frame render times and help mitigate this problem. It aims to, in one of a few ways, reduce render quality for parts of a VR scene in the periphery and/or increase render quality within the foveal view[10], exploiting the eye's space-variant resolution. Mohanto et al. [4] specify a range of subcategories for render quality reductions, all falling within 4 main subcategories: adaptive resolution (i.e. render the periphery with lower resolution or some other form of reduced sampling), geometric simplification (i.e. render objects with a less detailed model in the periphery), shading simplification/chromatic degeneration (i.e. simplify lighting simulation for pixels in the periphery), and spatio-temporal deterioration (i.e. partially reusing parts of the frame in the periphery to avoid having to re-render the entire view at each refresh).

To understand existing limitations, a high-level structure of a foveated rendering pipeline needs to be defined. There are two main types of foveated rendering pipeline with slightly different structures: static, and dynamic. Dynamic foveated rendering requires some form of eye tracking capability within the HMD to determine which part of the stereoscopic displays the

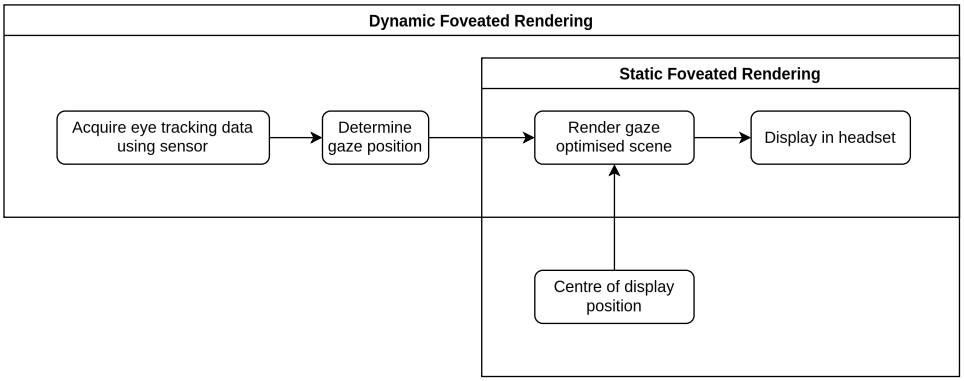

Figure 1: High level foveated rendering pipeline.

corresponding eye is directed towards. In software, frames must be received from this eye tracking device, and processed using some algorithm to determine where the eye is directed within a scene and/or on the physical display. The program being used must then be able to take this information and use it to in some way change how the scene is rendered. The rendered image must then be sent back to the headset to be displayed, all within a short enough timeframe to avoid noticeable visual latency. Static foveated rendering is mostly the same regarding the software implementation, however rather than requiring eye tracking hardware and processing, it simply assumes the fovea region is in the centre of each display and performs the rendering optimisations accordingly. This is not a realistic model, but will lead to performance gains at the cost of likely noticeable quality loss when the user averts their gaze from the centre. A summary of this pipeline can be viewed in fig. 1.

Static foveated rendering has been used by certain applications with non specialised hardware for years now. Dynamic foveated rendering has only started to become more common within the past few years, as increasingly mainstream HMDs such as PSVR2[**<empty citation>**] and Apple Vision Pro[**<empty citation>**] have been released with eye tracking functionality. The following sections aim to address why dynamic foveated rendering adoption is not yet ubiquitous across all HMDs, and identify shortcomings with existing implementations of both types of foveated rendering.

2 Time Limitations

Most limitations relate to the time taken to perform all of the foveated rendering process and optimisations. If this cannot be done in a shorter amount of time than just rendering an unchanged frame, then the entire process is redundant as visual latency has been worsened. Additionally,

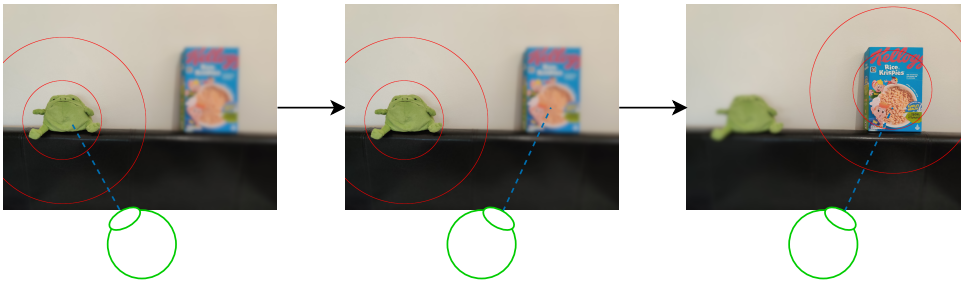

Figure 2: Large eye movements (saccades) result in the eye looking into the lower quality region before the frame updates. If eye tracking or image display is of high latency, this will be noticeable. Based on a similar diagram by Albert et al. [1].

with dynamic foveated rendering, if the foveated region cannot be updated with low enough latency, the issue shown in fig. 2 will occur where the user noticeably looks into the lower quality region of the scene after large, rapid eye movements (saccades). In this section, issues will be broken down in the order of the dynamic foveated rendering pipeline as defined in fig. 1. Difficulties faced will be similar for static foveated rendering, ignoring parts relating to eye tracking.

The first time-based (hardware) consideration is the time taken to acquire sensor data for eye tracking. TODO FINISH

The second time-based (software) consideration is regarding the processing of eye tracking data from a sensor to determine gaze position. This is often performed using a trained machine learning model. TODO FINISH

The third time-based (based) consideration is regarding the gaze optimised rendering of the scene. TODO FINISH THIS BIT

The final time-based (based) consideration is regarding displaying the image in the headset. Since this must be performed even when not performing foveated rendering it presumably is not a source of problems. I was unable to find any literature referencing foveated rendering specific difficulties with this step (CHECK THIS!!). TODO FINISH THIS BIT

3 Other Software Limitations

A significant hardware limitation is simply adoption. Since only very few headsets currently implement eye tracking of any sort, there is little motivation for developers to implement foveated rendering support if there is not a significant enough user base.

4 Other Hardware Limitations

A significant hardware limitation is simply adoption. Since only very few headsets currently implement eye tracking of any sort, there is little motivation for developers to implement foveated rendering support if there is not a significant enough user base.

5 User Perception

6 Conclusion

References

[1] Rachel Albert et al. "Latency requirements for foveated rendering in virtual reality". In: *ACM Transactions on Applied Perception (TAP)* 14.4 (2017), pp. 1–13.

[2] Detlev Arendt. "Evolution of eyes and photoreceptor cell types." In: *International Journal of Developmental Biology* 47.7-8 (2003), pp. 563–571.

[3] Marc Levoy and Ross Whitaker. "Gaze-directed volume rendering". In: *Proceedings of the 1990 symposium on interactive 3d graphics*. 1990, pp. 217–223.

[4] Bipul Mohanto et al. "An integrative view of foveated rendering". In: *Computers & Graphics* 102 (2022), pp. 474–501.

[5] Richard J Pumphrey. "The theory of the fovea". In: *Journal of Experimental Biology* 25.3 (1948), pp. 299–312.

[6] Kjetil Raaen and Ivar Kjellmo. "Measuring latency in virtual reality systems". In: *Entertainment Computing-ICEC 2015: 14th International Conference, ICEC 2015, Trondheim, Norway, September 29-Ocotober 2, 2015, Proceedings 14*. Springer. 2015, pp. 457–462.

[7] Thomas Waltemate et al. "The impact of latency on perceptual judgments and motor performance in closed-loop interaction in virtual reality". In: *Proceedings of the 22nd ACM conference on virtual reality software and technology*. 2016, pp. 27–35.

[8] Lili Wang, Xuehuai Shi, and Yi Liu. "Foveated rendering: A state-of-the-art survey". In: *Computational Visual Media* 9.2 (2023), pp. 195–228.

[9] Cornelius Weber and Jochen Triesch. "Implementations and implications of foveated vision". In: *Recent Patents on Computer Science* 2.1 (2009), pp. 75–85.

[10] Martin Weier et al. “Foveated real-time ray tracing for head-mounted displays”. In: *Computer Graphics Forum*. Vol. 35. 7. Wiley Online Library. 2016, pp. 289–298.