Measuring Latency in Virtual Reality Systems

Kjetil Raaen[1,2,3] and Ivar Kjellmo[1]

[1] Westerdals - Oslo School of Arts, Communication and Technology, Oslo, Norway
[2] Simula Research Laboratory, Bærum, Norway
[3] University of Oslo, Oslo, Norway

Abstract. Virtual Reality(*VR*) systems have the potential to revolutionise how we interact with computers. However motion sickness and discomfort are currently severely impeding the adoption. Traditionally the focus of optimising VR systems have been on frame-rate. Delay and frame-rate are however not equivalent. Latency may occur in several steps in image processing, and a frame-rate measure only picks up some of them. We have made an experimental setup to physically measure the actual delay from the user moves the head until the screen of the VR device is updated. Our results show that while dedicated VR-equipment had very low delay, smartphones are in general not ready for VR-applications.

Keywords: Latency, Virtual Reality, Framerate, Mixed Reality.

1 Introduction

The last years have seen a great increase in interest and popularity of Virtual Reality (VR) solutions. Some are dedicated hardware made specifically for VR, such as Oculus Rift. Others are Mobile VR solutions such as Samsung's Gear VR, HTC and Valves' new Steam VR and even the simplified solution Google Cardboard. Providing a stereoscopic view that moves with their head, these systems give users an unprecedented visual immersion in the computer generated world.

However, these systems all have in common the same potential problems when it comes to motion sickness and discomfort when using the VR solutions[2]. An important source of these problems is delay. This paper presents an apparatus for measuring delay as well as detailed measurements of delay in some popular VR systems.

Frame-rate is a measurement of how fast frames are sent through the rendering pipeline. Delay on the other hand defines the time it takes from a user triggers an action, such as turning the head, until results are visible on screen. Previous work by one of us[7] has, however indicate that there is no linear relationship between frame-rate and delay in traditional computer graphics setups. From this, we assume that frame-rate is not the best metric. If VR solutions work like other graphics output devices, there is a significant difference between frame-rate measured inside an application and actual update speed.

K. Chorianopoulos et al. (Eds.): ICEC 2015, LNCS 9353, pp. 457–462, 2015.
DOI: 10.1007/978-3-319-24589-8_40

By using a light sensor and an oscilloscope we hope to measure the exact delays in VR applications. Finding out how much delay there is in virtual reality headsets would be helpful in when investigating the causes for motion sickness in virtual worlds, as well as providing guidance to people producing software for these systems.

2 Background

Previous research has suggested various methods for measuring delay in VR systems all the way back to the previous VR-hype in the early nineties. Earlier work has used cameras [6] or light sensors [8]. What they all have in common is that they rely on a continuous, smooth movement of the tested devices. DiLuca et al.[3] summarise a series of them, and suggest their own approach. Their approach is the most elegant we have found so far.

2.1 Sources of Delay

This section discusses the parts of the VR-pipeline that add most to the response delay. Components in the pipeline from the time the user moves until displays update are in general *black boxes*; documentation about how they work is often lacking. Thus, the only way to evaluate these delays is by measuring.

Screens used to display output add some delay, which can be divided into two parts: *screen refresh* and *response time*. LCD screens used in VR displays receive and display updated images at a fixed rate. This rate is termed *screen refresh* rate. Most modern screens update at 60 frames per second (FPS), or every 16.7 ms. *Response time* denotes the time the physical pixels take to change colour.

A *frame buffer* is memory used for holding a rendered frame about to be displayed on the monitor or VR-display. Modern renderers use at least two frame buffers. To avoid showing unfinished frames to the user, drawing happens in one buffer, while the other is being displayed. This practice is termed *double buffering*. Further, to avoid showing parts of two different buffers, it is common to wait for the next screen refresh before swapping. When double buffering is used, rendering follows the sequence: Assume frame 0 is the frame during which an event from an input device is registered. Frame 1 contains the result of the event, and at the time of frame 2 the result is sent to screen. This gives a minimum of 1 full frame time from input event to result on screen.

At 60 FPS this adds up to a minimum of one frame delay (17 ms) to a maximum of two frames (33 ms) delay. Further, not all hardware is capable of keeping up with the target frame-rate at all times. Slow hardware leads to significantly longer delays. An increased number of frame buffers in the pipeline increases this delay, because more steps are added between rendering and displaying data.

Dedicated VR equipment has tailor-made hardware and drivers that have been tweaked for low latency performance. The gyros of smartphones on the other hand are mainly intended for navigation or keeping the screen rotated the correct way, neither of which require fast response time or high accuracy.

2.2 Acceptable Delay

Empirical values for how long delay is acceptable is difficult to find. Davis et al. investigated a condition they term *cybersickness*[2] in analogy to motion sickness. They consider a range of options for the cause of these problems, delay among them. However they do not quantify delay that might lead to symptoms. Jarods [5] concludes that people reacts very differently to latency. Some people would hardly notice a 100 ms delay while other very sensitive people are able to perceive down to 3-4ms of latency.

Most design guidelines and measurements reports are from developers and of VR equipment and VR-software rather than scientists. John Carmack [1] states that a latency of 50 ms feels responsive but the lag when moving in the virtual world is noticeable. He recommends that latency should be under 20 ms. Neither methodology or background numbers and test results are presented.

3 Experiment Setup

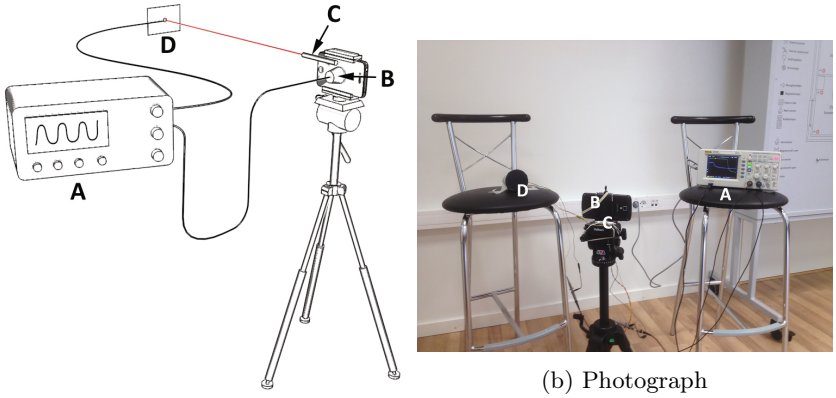
To measure response time after abrupt movements, we need the virtual reality device to run a program that detects a small rotation and as fast as possible change the displayed picture. A simple program that detects rotation and changes the displayed picture would create solve this problem. However, we are interested in delays from actual 3D virtual reality software. Therefore we used a popular game engine, Unity 3D[1], and made a scene that creates abrupt changes based on headset movement. We tested multiple VR devices. Oculus Rift is a dedicated VR display solution, designed for this purpose. The other systems are smartphones, which developers have discovered have all the required hardware to run VR application and can also function as headsets.

The physical setup(fig 1) consists of the VR device (Oculus Rift, Smartphone etc.) mounted on a camera tripod. One light sensor is attached to the screen of the VR-device to register the virtual scene shifting from white to black. A laser pen is also attached to the virtual device pointing at another light sensor approx. one meter from the tripod setup. Both light sensors are connected to an oscilloscope. When we move the VR device by turning the tripod, the light sensor illuminated by the laser pen registers the disappearance of the light. When the movement is detected, light sensor connected to the virtual device screen measures the light shift from the white plane disappearing in the virtual scene.

This allows us to measure the time from when the physical movement starts until in movement is visible in the virtual scene. We measured 5-10 times on each device and we present an average of the measured values.

The virtual setup running on the VR-device is a simple completely black virtual 3D scene with a white self-illuminated plane and a virtual stereo camera. The virtual camera is set up with normal VR movement controllers. For the mobile phone setup the Durovis Dive SDK[2] was used while for the Oculus rift setup

[1] `http://unity3d.com`
[2] `http://www.durovis.com/sdk.html`



(a) Schematic drawing

(b) Photograph

Fig. 1. The Physical setup with the Oscilloscope (A), light sensor (B) attached to Virtual device, Laser pen (C) and light sensor (D) picking up the laser.

the Oculus Rift SDK[3] was used. This in order to set up a virtual reality scene with an absolute minimum of content optimised for a highest possible frame-rate, giving consistent values of multiple hundred frames per second and to use the build in VR movement controllers. The OculusRift devices were connected to a fast laptop[4].

The virtual scene consists of a white plane on a black background. The camera faces the white plane from a large distance and is set to a very narrow field of view. The scene is tweaked so that the display is completely white initially. Because of narrow field of view the camera is very sensitive to rotation. This means that even a small rotation in the headset leads to the screen changing colour from white to black, a change picked up by the light sensor.

4 Results and Discussion

Measured delay is simply the time from the light sensor illuminated by the laser changes from bright to dark until the light sensor attached to the screen reports the same change. Table 1 shows the results from the systems we tested. Unsurprisingly, the dedicated hardware has much faster response rate. Further, the smartphones do not give developers access to control the vertical synchronisation setting, and it seems from the numbers that it is always on.

For the dedicated VR displays, v-synch has a large effect on total delay. While the Virtual scenes with V-sync off the frame-rate in the application was as high as 3000 fps, the one with V-sync on would hold a steady frame-rate at 60 fps. Turning off this feature introduces introduce visual artefacts, but reduces the

[3] `https://developer.oculus.com/`

[4] Windows 7, Intel i7 - 3740 QM CPU @ 2,70 ghz, NVIDIA Quadro K2000M - 2048 mb DDR3

Table 1. Results from delay measurements.

VR Display	Avg.	Min.	Max.
Oculus Rift dev kit 1, v-sync ON	63 ms	58 ms	70 ms
Oculus Rift dev kit 1, v-sync OFF	14 ms	2 ms	22 ms
Oculus Rift dev kit 2, v-sync ON	41 ms	35 ms	45 ms
Oculus Rift dev kit 2, v-sync OFF	4 ms	2 ms	5 ms
Samsung Galaxy S4(GT-I9505)	96 ms	75 ms	111 ms
Samsung Galaxy S5	46 ms	37 ms	54 ms
iPhone 5s	78 ms	59 ms	96 ms
iPhone 6	78 ms	65 ms	91 ms

delay significantly. Without vertical synchronisation, these products can react very fast. Smartphones on the other hand are much slower, with delays close to 100 ms for most models. The exception is the Samsung S5 which has a delay less than 50ms. This result is similar to the Oculus Dev kit 2 with V-sync on. The screen in the Samsung S5 is actually the same as in the Oculus Dev kit 2 and the result confirms the similarity between the two devices.

These results come with some caveats. Despite our efforts, the rendered scene might not turn instantly from white to black between one frame and the next. Intermediate frames should show up as plateaus in the oscilloscope output, but we cannot be completely sure the animated movement did not show a frame or a few of parts of the white square. However, consistent results indicate that this is unlikely.

5 Conclusions and Future Work

We have presented a simple and precise solution for measuring delay in VR-systems. In contrast to DiLuca's [3] and other previous work, our system are able to detect delays in instantaneous, jerky movements. Earlier systems have generally relied on smooth, continuous movements, which are quite alien to a real user in a chaotic game. Jerald finds in 2012 [4] that users are most sensitive to delays when their movement changes direction. Current and future VR technology often employ prediction to reduce delay during continuous motion, which does work when motion changes. Thus, measuring delay for these sudden movements produces results more aligned with the sensitivity of users. Our setup does not require any modifications to the VR-systems to measure their delay, and can thus easily be applied to new hardware as soon as it becomes available.

Regarding how short delay is *good enough* both developer guidelines and than scientific papers mostly agree, placing ideal delay at less than 20 ms[4]. Even if the numbers comes from a simulated environment it gives a quality study of human perception when it comes to noticing latency.

With this in mind, it seems clear that the phones are too slow. Both of the latest models of iPhone, as well as the older Samsung have latencies around 100 ms, which all sources agree is far too slow. The older phones are not designed to run

VR systems, but these results give us a picture of the difference of yesterday's standards and the present specifications needed for pleasant VR experiences on mobil. In this picture the Samsung Galaxy S5 is closer to acceptable. Depending on which limits you use, an average value of 46 ms is either barely acceptable or somewhat too slow. However, Samsung markets some phones clearly as VR-devices, such as Samsung Note 4 and the upcoming Samsung Note 5. These are currently the only devices working with Gear VR, their new system for combining phones with VR. Our results from the Galaxy S5 indicates that some optimisations for VR are already included in this phone.

Oculus Rift in both versions respond extremely fast with v-synch off, fast enough to satisfy most guidelines. With synchronisation on, on the other hand, they are barely fast enough. The same guidelines that recommend 20 ms delay also claim v-synch is important for user experience. We found no way of satisfying both the delay requirement and the requirement to use vertical synchronisation at once. Oculus Rift is advertised to contain a built in latency tester. We did not find documentation on how to use the, nor did we find any description on how it works. Comparing our results with the output of this hardware would be interesting.

Development of new VR technologies, both dedicated and mobile, is progressing at a rapid pace these days, and not all the systems presented here will be current by the time this paper is published. The experiment setup on the other hand should be useable for any new system, and we hope to update our data with new systems when they become available. Other factors influencing user experience should also be studied in more detail, as well as their interaction with delay.

References

1. Carmack, J.: Latency Mitigation Strategies (2013), https://www.twentymilliseconds.com/post/latency-mitigation-strategies/
2. Davis, S., Nesbitt, K., Nalivaiko, E.: A Systematic Review of Cybersickness. In: IE 2014. ACM, New York (2014)
3. Di Luca, M.: New Method to Measure End-to-End Delay of Virtual Reality. Presence: Teleoperators and Virtual Environments 19(6), 569–584 (2010)
4. Jerald, J., Whitton, M., Brooks, F.P.: Scene-Motion Thresholds During Head Yaw for Immersive Virtual Environments
5. Jerald, J.J.: Scene-Motion- and Latency-Perception Thresholds for Head-Mounted Displays PhD thesis (2009)
6. Kijima, R., Ojika, T.: Reflex HMD to compensate lag and correction of derivative deformation. In: Proceedings IEEE Virtual Reality 2002 (2002)
7. Raaen, K., Petlund, A.: How Much Delay Is There Really in Current Games? In: ACM MMsys, pp. 2–5 (2015)
8. Swindells, C., Dill, J., Booth, K.: System lag tests for augmented and virtual environments. In: Proceedings of the 13th Annual ACM . . . , vol. 2, pp. 161–170 (2000)