How much delay is there really in current games?

Kjetil Raaen
Westerdals — Oslo School of Arts,
Communication and Technology
1325 Lysaker, Norway
Simula Research Laboratory
University of Oslo
raakje@westerdals.no

Andreas Petlund
Simula Research Laboratory
1325 Lysaker, Norway
apetlund@simula.no

ABSTRACT

All computer games present some delay between human input and results being displayed on the screen, even when no networking is involved. A well-balanced discussion of delay-tolerance levels in computer games requires an understanding of how much delay is added locally, as well as in the network. This demonstration uses a typical gaming setup wired to an oscilloscope to show how long the total, local delay is. Participants may also bring their own computers and games so they can measure delays in the games or other software.

Results show that local delays constitute such a large share of the total delay that that it should be considered when studying the effects of delay in games, often far exceeding the network delay evaluated.

1. INTRODUCTION

Researchers have worked on how delay influences players in networked games for some years now [1, 3, 5, 8, 10, 14]. This work has traditionally focused on pure network delay in multiplayer games. In these games, latency is somewhat alleviated by locally echoing user actions without waiting for server confirmation, as well as dead reckoning of other players postitions [6]. Recently there has been a trend towards studying delays in cloud gaming [11,13]. These delays are somewhat different, because they occur directly between the user's input and the observed output. No techniques are currently implemented to hide this latency.

What is lacking from from all these publications is a consideration of the local system delay. All of these studies report the network latency. The network latency studied may be from actual network conditions or artificially induced. What is missing from the latency analysis are reports on how much local delay is present in the test setup. Discussions on the influence of local latency for the player tolerance and experience is also missing. Such local latency can constitute a large part of the total latency and vary considerably depending on the hardware used and software configurations.

http://dx.doi.org/10.1145/2713168.2713188
MMSys '15, Mar 18-20, 2015, Portland, OR, USA

Table 1: Some monitors and their response time.

Brand	Type	Response time
Apple	Thunderbolt Display	12 ms [2]
Dell	UltraSharp 24	8 ms [12]
Asus	ROG SWIFT PG278Q	1 ms [4]

Therefore, knowing what share the network and local delays constitute is important for a proper analysis of the effect of one of the components on user experience.

The goal of this work is to investigate and demonstrate how much total system lag is present in current computer systems used for gaming. We will allow participants to measure delays in their own hardware and software. Further, we propose a method for measuring such delay in experiments seeking to quantify individual components of gaming delay. Lastly, we will compare this delay to a similar setup using a cloud gaming service instead of local rendering.

2. IMPORTANT COMPONENTS IN LOCAL DELAY

Even in a local game with no networking, there is potential for delays at levels perceptible to humans, or affecting human task performance. This section discusses the parts of the pipeline that add most to the response delay in the local system. Components in the pipeline from user input to screen respones are to a large degree *black boxes*; documentation about how they work is often lacking. Thus, the only way to evaluate these delays is by measuring.

2.1 Monitors

Screens used to display output add some delay, which can be divided into two parts.

2.1.1 Monitor update

LCD screens 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. Some screens specialised for gaming or other response critical applications can update much faster.

2.1.2 Monitor response time

Monitors vary wildly in response time, that is the time they take from one shade of grey to another, from 1 ms for screens designed for gaming, to 12 ms for typical office screens, see table 1 for some examples. The exact procedure and color values for this test are not standardised, and it is

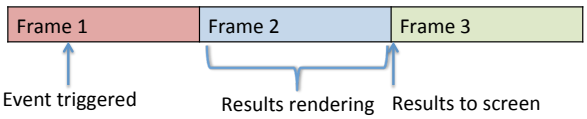

Figure 1: Timeline for events with double buffering.

reasonable to assume that manufacturers choose the conditions most favorable to their product.

2.2 Frame buffers

A frame buffer is memory used for holding a rendered frame about to be displayed on the monitor. Modern games use at least two frame buffers. To avoid showing unfinished frames on the screen, 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. The terms *vertical synchronization* or *vsynch* are used, because historically the swap was performed when the electron beam was returned to the start of the frame, a period of time called the *vertical blanking interval*.

When double buffering is used, rendering follows the sequence: An event from an input device is registered during frame N, frame N+1 contains the result, and at the time of frame N+2 the result is sent to screen, as shown in figure 1. This gives a minimum of 1 full frame time from input event to result on screen. At 60 FPS this adds up to a total of 17 - 33 ms delay. Many games have a target framerate of only 30 FPS. At this rate, the delay from screen refresh and frame buffer pipeline is 33 - 67 ms. Further, not all hardware is capable of keeping up with the target framerate at all times. Slow hardware will lead to significantly longer delays. An increased number of frame buffers in the pipeline will increase this delay, because more steps are added between rendering and displaying data. High system load from the game itself or external tasks can lead to lower frame-rate, and thus add to this number.

2.3 Input device latency

The best gaming equipment has tailor-made drivers that have been tweaked for low latency performance. A good gaming mouse may add ∼2 ms [15] to the latency. Devices that are not made for gaming may add more.

3. DEMO SETUP

Measuring delays between input and output accurately requires a purpose-built setup. Previous, informal work as described in section 5 measured delay by filming the screen and a button with a high-speed camera while pushing the button repeatedly. This approach has some limitations. First, it is limited to the capture rate of the camera. Secondly, it is difficult to judge exactly when the button was pushed, and lastly it requires tedious manual work to analyse the videos.

3.1 Exact measurements

To get more exact results this demonstration will use an oscilloscope to measure the timing difference, see figure 2. The left most button is connected to the oscilloscope using

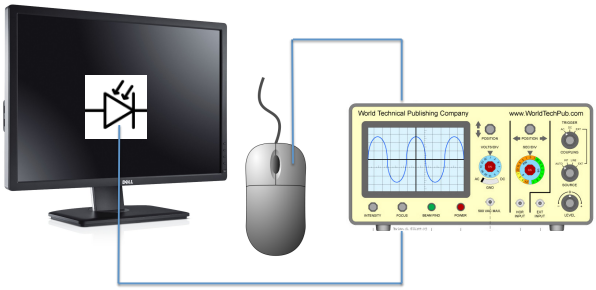

Figure 2: Sketch of demo setup.

Figure 3: Capture of trigger event. Yellow line represent output from the mouse, blue line represents output from the photosensor. Both are active low. The difference in time between the two flanks are shown in green in the bottom-left corner of the oscilloscope screen.

an additional wire. A photosensor is held on the screen, and gives a signal when something bright appear. These signals are fed to the two channels of the oscilloscope. The participants find a dark place in the game. Then can they trigger an in-game effect that lights up the screen fast. Firing a weapon will in most games achieve this. Participants can thus see the delay between the two evens with high accuracy, by comparing the flank representing the mouse click (yellow on figure 3) with the flank representing change in light from the screen (blue on firgure 3).

Our setup is independent of the hardware and software used to play the game. This allows testing real, commercial games, without modifying software or hardware. Results will represent the sum total of all delays. Delay from the input device, the game software, graphics card drivers, the graphics card itself or the screen are all added up. Investigating how much delay each part of the test system introduce requires modifying the test system itself locking the demo setup to one pre-modified testbed rig. We want participants to be able to test the delay of computers they bring to the setup to document the differences shown by a wide range of setups.

Table 2: Results from experiments.

Experiment	Avg.	Min.	Max.
Rectangle, vsync on	82 ms	73 ms	102 ms
Rectangle, vsync off	27 ms	21 ms	41 ms
UT3, default, vsync off	58 ms	52 ms	66 ms
UT3, default, vsync on	95 ms	79 ms	102 ms
UT3, optimised, vsync off	33 ms	23 ms	38 ms

3.2 Cloud games

To test cloud games, we will present exactly the same setup, but participants play the game remotely through a commercial cloud gaming service. In parallel we will also show the network latency to the cloud gaming provider in realtime, so participants can see how large proportion of the delay is due to network latency. This part is contingent on access to cloud gaming services at the venue of the demonstration.

4. PRELIMINARY RESULTS

To evaluate how well this setup works, and highlight how the results can be useful, we ran some preliminary experiments. For each condition we repeated the test 10 times, reporting minimum, maximum and average results. We used only one system for the purposes of this demo, planning a more exhaustive analysis of a wide range of systems in future work. This system was a MacBook Pro (Retina, 15-inch, Early 2013)[1], running OSX for the rectangle demo and Windows 7 for the game.

First, we ran a simple program that renders a rectangle in OpenGL in response to mouse-clicks. We compare this program running with and without vertical synchronisation. With vertical synchronisation it ran at about 60 FPS, without vsync the framerate fluctuated between 400 and 700 FPS.

Next we ran the game *Unreal Tournament 3* (UT3) and timed a basic pistol shot, an event that is supposed to be *immediate*. For this game we tested using three different conditions. Firstly, we tested using default settings. These had vsync off and a resolution of 1024 by 768. At these settings the game ran at about 60 FPS. Then we simply turned on vertical synchronisation and ran the experiment again, now the framerate was stable at exactly 60 FPS. Lastly we ran without vsync and with all settings optimised for faster framerate. These kind of tweaks are popular with professional gamers.

As we see in table 2, response time is highly variable, even in the same game running on the same setup. Average delays in UT3 vary from 33 ms to 95 ms depending on the settings.

5. RELATED WORK

We have not been able to find any references to scientific publications measuring the local system latency for games. The community for hardware component benchmarking and game optimisation, however, have touched on the topic on occasion.

Rendering Pipeline [15] finds values of between 76 ms and 112 ms for the game *Half Life 2* using realistic settings. *Blur Busters*, test multiple games [7]. They get results of 72–77 ms for *Battlefield 4*, 52–59 ms for *Crysis 3* and 22 ms to 40 ms for *Counter Strike: Global Offense*.

These numbers indicate that different games handle input in very different ways. Also, we know that network delays as low as 50 ms affects player performance in some games [9]. It is therefore important to investigate how player performance is affected by local delays of similar magnitudes.

[1]2.4 GHz Intel Core i7, 16 GB 1600 MHz DDR3, NVIDIA GeForce GT 650M

6. CONCLUSIONS

When evaluating the effect of latency on the perceived performance of game systems, it is important to consider that the network delay is added to the delay of the local system. Preliminary results from this demo show delays of up to 100 ms. This shows that not only is the local delay at the same order of magnitude as network delays, it might in many cases be *larger*.

When discussing the tolerance levels for latency in networked games, both cloud gaming and traditional games, care should be taken to know the local delay as it may vary greatly between platforms depending on both hardware and software, in addition to software configuration.

This local latency may influence the measured tolerance levels in QoE experiments and bias the numbers reported as tolerance levels. It is therefore important that studies of the effects of other types of delay measure the inherent system delay when presenting numbers on how delays affect players. We propose using the setup presented in this demo to calibrate such experiments. Without such measurements it is very difficult to compare results from different studies on response time requirements.

In the future, we would like to investigate an array of different games and hardware under varied conditions. This will increase our understanding of delays in local gaming systems, and could likely be important in related fields such as Human-Computer Interaction (HCI).

Acknowledgements

The work in this paper was part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700) and part funded by the Research Council of Norway under the "TimeIn" project (No: 213265). The views expressed are solely those of the author(s).

7. REFERENCES

[1] Amin, R., Jackson, F., Gilbert, J., Martin, J., and Shaw, T. Assessing the Impact of Latency and Jitter on the Perceived Quality of Call of Duty Modern Warfare 2. In *Human-Computer Interaction. Users and Contexts of Use*, M. Kurosu, Ed., vol. 8006 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 97–106.

[2] Apple. Apple Thunderbolt Display.

[3] Armitage, G. An experimental estimation of latency sensitivity in multiplayer Quake 3. In *The 11th IEEE International Conference on Networks 2003 ICON2003* (2003), IEEE, pp. 137–141.

[4] Asus. Asus ROG SWIFT PG278Q.

[5] Beigbeder, T., Coughlan, R., Lusher, C., Plunkett, J., Agu, E., and Claypool, M. The

effects of loss and latency on user performance in unreal tournament 2003. In *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games* (New York, NY, USA, 2004), NetGames '04, ACM, pp. 144–151.

[6] Bernier, Y. Latency compensating methods in client/server in-game protocol design and optimization. *Game Developers Conference 98033*, 425 (2001).

[7] BlurBusters. Preview of NVIDIA G-SYNC.

[8] Claypool, M. The effect of latency on user performance in Real-Time Strategy games. *Computer Networks 49*, 1 (Sept. 2005), 52–70.

[9] Claypool, M., and Claypool, K. Latency and player interaction in online games. 40–45.

[10] Claypool, M., and Claypool, K. Latency Can Kill : Precision and Deadline in Online Games. *Proceedings of the First ACM Multimedia Systems Conference* (2010).

[11] Claypool, M., and Finkel, D. The Effects of Latency on Player Performance in Cloud-based Games. In *NetGames 2014(in print)* (2014).

[12] Dell. Dell UltraSharp 24 Ultra HD Monitor.

[13] Jarschel, M., Schlosser, D., Scheuring, S., and Hoss feld, T. An Evaluation of QoE in Cloud Gaming Based on Subjective Tests. *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (June 2011), 330–335.

[14] Li, S., Chen, C., and Li, L. Evaluating the Latency of Clients by Player Behaviors in Client-Server Based Network Games. *2008 3rd International Conference on Innovative Computing Information and Control* (2008), 375–375.

[15] Measuring Input Latency. Measuring Input Latency.