Visual stimulus timing precision in Psychtoolbox-3
Tests, pitfalls & solutions

MAX-PLANCK-GESELLSCHAFT

Mario Kleiner (mario.kleiner@tuebingen.mpg.de) Max Planck Institute for Biological Cybernetics, Tübingen, Germany

MPI FOR BIOLOGICAL CYBERNETICS

Introduction

Visual stimulation paradigms in perception research often require accurate timing and timestamping for presentation of visual stimuli.

This work tested the visual stimulus onset timing and timestamping precision under Psychtoolbox (PTB) [http://www.psychtoolbox.org] on a few different systems. It also exemplifies the influence of the chosen scheduling strategy under different system loads and of the graphics cards dynamic power management on the accuracy of visual stimulus onset timing.

Stimulus presentation in PTB

Stimulus images are drawn into a hidden offscreen backbuffer. Stimulus onset is requested at a specific target system time *twhen*. PTB tries to display the stimulus image at start of the first video scanout cycle after *twhen*. It returns an estimated timestamp *tvbl* of true stimulus onset:

tvbl = Screen('Flip', window, twhen);

Example animation loop: Show new image *waitframes* video refresh cycles after last image (presented at *tvbl*).

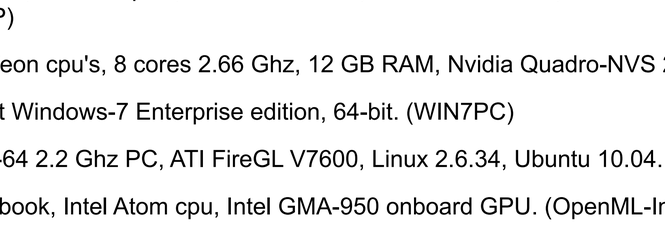
ifi=Screen('GetFlipInterval', window);

for i=1:nImages

Draw i'th new stimulus image ...

tvbl = Screen('Flip', window, tvbl + ifi(waitframes – 0.5));*

end



Test setup and protocol

Display: Sony GDM-F500R CRT, 1024 x 768 pixels at 60 Hz refresh.

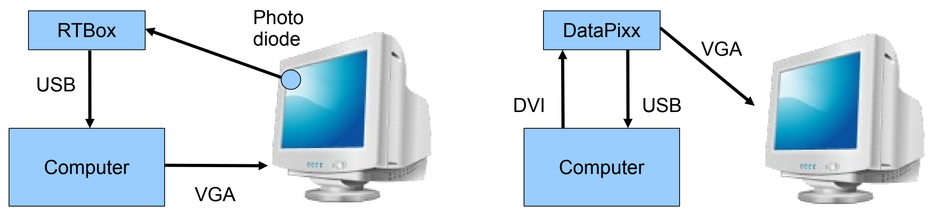
External stimulus onset measurement:

* Photodiode + RTBox device + PsychRTBox driver (p)[http://lobes.usc.edu/RTbox/]

* Vpixx DataPixx box + PsychDataPixx driver (d)[http://www.vpixx.com]

Tested operating systems and graphics cards:

* Dell Optiplex-960 PC, Core2-Duo 3.0 GHz, Radeon HD 3470:
  - Microsoft Windows Vista Business edition, 32-bit, SP1. (VISTAPC)
  - Ubuntu Linux 9.10, standard Linux kernel 2.6.31. (LNXPC)
  - Ubuntu Linux 9.10, realtime Linux kernel 2.6.31-9 RT. (RTLNXPC)
* Apple MacBook Pro, Core2-Duo 2.4 Ghz, Geforce 8600M GT:
  - Microsoft Windows-7 Enterprise edition, 32-bit. (WIN7MBP)
  - Ubuntu Linux 9.10, realtime Linux kernel 2.6.31-9 RT. (RTLNXMBP)
  - Mac OS/X Snow Leopard 10.6.2. (SLMBP)
* Apple Mac Pro, 2 Xeon cpu's, 8 cores 2.8 GHz, GeForce 8800GT, Mac OS/X Leopard 10.5.8. (LPMP)
* Dell PC, 2 Xeon cpu's, 8 cores 2.66 Ghz, 12 GB RAM, Nvidia Quadro-NVS 295:
  - Microsoft Windows-7 Enterprise edition, 64-bit. (WIN7PC)
* AMD Athlon-64 2.2 Ghz PC, ATI FireGL V7600, Linux 2.6.34, Ubuntu 10.04. (OpenML-ATI)
* HP-Mini Netbook, Intel Atom cpu, Intel GMA-950 onboard GPU. (OpenML-Intel)
* Dell Optiplex GX620 PC, Pentium-4 hyperthreading cpu, 2 logical cores, 3.0 GHz, Geforce 7800 GTX GPU under Windows XP Pro SP3, 32-bit. (WINXPPC)



Sequence of 9953 visual timing trials with different repeated interstimulus intervals:

1. Block: Simulation of constant framerate animation with 4972 trials.

2. Block: Repeat 4972 trials from 1st block, but in randomly permutated order to simulate complex presentation schedules.

3. Block: Accuracy over long interstimulus intervals (5 secs, 10 secs, 20 secs).

* Simultaneous sound playback and USB activity to create additional interrupt load.

* 50-75% variable cpu load, 50% rendering load on gpu to simulate experiment workload.

Repetitions:	1800	1200	450	360	300	257	225	200	180	3	3	3
TargetISI in Frames:	2	3	4	5	6	7	8	9	10	300	600	1200

Results: Stimulus onset precision

A trial is counted as „skipped frame“ if it has a measured onset time that is more than 0.75 video refresh durations later than expected, according to presentation schedule.

Impact of presentation method (synchronous Flip vs. asynchronous Flip) and realtime scheduling strategy under low and high system load:

Load:	No external load: System otherwise idle				Medium external load: 50% of all processor cores busy			
Flip method:	Sync	Sync-RT	Async	Async-RT	Sync	Sync-RT	Async	Async-RT
VISTAPC	1 / 1	4 / 4	1 / 1	2 / 2	3675 / 3599	1 / 1	2 / 1	1 / 1
LNXPC	1 / 1	2 / 2	0	6 / 6	374 / 265*	4 / 4*	401 / 385	662 / 646*
RTLNXPC	0	0	0	0	11 / 11*	0	30 / 30*	0
LPMP	16 / 16	3 / 3	18 / 18*	3 / 3	487 / 487*	40 / 40*	477 / 477*	121 / 121*
WINXPPC	7 / 4	53 / 53*	8 / 3	3 / 2	failed	2707 / 2707	failed	crash

Counts are #total misses / #misses for 1 frame ISI. E.g., „374 / 265“ means: Total of 374 skipped frames, 265 skipped for short presentation deadlines of one video refresh interval. A (*) means that frame skips happened mostly during 2nd block of trials with permutated ISI's.

* Synchronous *Screen('Flip')* and asynchronous *Screen('AsyncFlipBegin')* flips work equally well.

* Use of realtime scheduling (*Priority()*) helps significantly, especially under system load.

* Realtime Linux was only system that performed well even under 100% external system load.

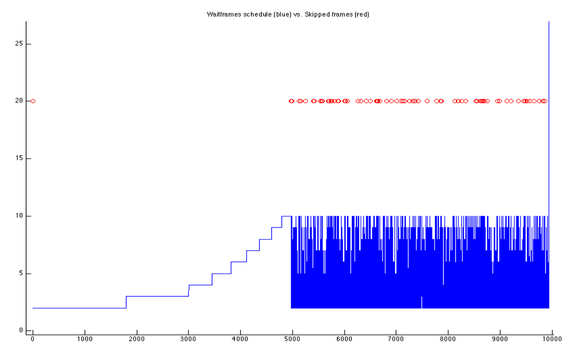
* Skipped presentation deadlines happen mostly for 1 frame interstimulus intervals, almost never for longer intervals. Suggests stimulus onset scheduling is accurate as long as gpu can handle workload.

* No skipped deadlines observed for interstimulus intervals of 5 – 20 seconds.

Sync-RT only:	VISTAPC	LNXPC	RTLNXPC	WINXPPC	LPMP	SLMBP	WIN7MBP	WIN7PC
0% external load:	4	2	0	53	3	2	126	22
50% external load:	1	4	0	2707	40	181	1594	111
100% external load:	1219,tearing	1826	0	Failed	470	2079	Failed	Failed

* Beware of dynamic GPU power management!
Can affect precision of presentation schedule and cause skipped deadlines, especially...
... when schedule is irregular
... stimulus is not very demanding
... stimulus onset is triggered by external events, e.g., subject's response, external equipment.

- *PsychGPUControl()* or system specific tools to disable dynamic GPU power management.
- *Screen('DrawingFinished');* may help.



Results: Stimulus timestamp precision

PTB uses the video scanout position of a display device to remove timing variability (caused by random system latencies) from swap completion timestamps, and to remap timestamps to correspond to the start of video scanout at the top of a display. Thereby it can provide precise stimulus onset timestamps.

Example: Raw timestamp variability on MS-Vista under external system load.
Mean 3.413 ms Max10.581 ms

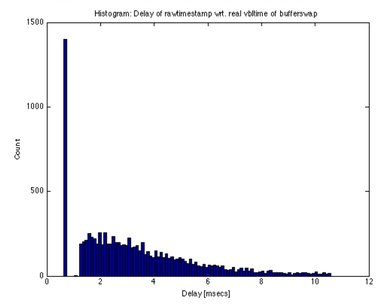
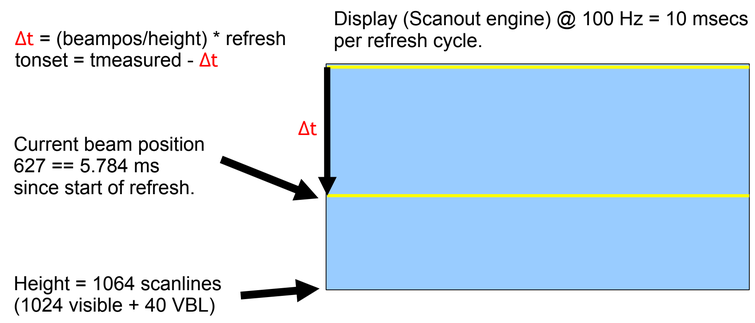


Illustration of working principle:



Method:	Error of high precision timestamping (in ms):			Error of raw (naive) timestamping (in ms):			
System:	MeanDiff	Stddev:	Range:	MeanDiff	Stddev:	Range:	Caveat:
(d)-WIN7PC	0.783	0.022	0.228	0.036	0.132	12.095	Bias?Beampos-workaround.
(p)-WIN7MBP	1.236	0.090	0.921	1.221	0.185	10.552	Bias?
(p)-SLMBP	0.734	0.060	0.351	0.691	0.060	0.351	Beamposwork-around needed.
(p)-RTLNXMBP	n/a	n/a	n/a	1.329	0.038	1.116	-
(p)-WINXPPC	0.562	0.068	0.564	0.796	1.424	16.553	Display corruption on overload.
(p)-VISTAPC	0.935	0.065	0.757	0.871	0.187	13.305	-
(p)-LNXPC:	1.160	0.073	0.534	1.103	0.544	16.162	Bias ~ 0.74 ms.
(p)-RTLNXPC:	1.235	0.075	0.554	1.229	0.107	2.186	Bias ~ 0.74 ms.
(p)/(d)-LPMP:	0.657 / 0.115	0.069 / 0.023	0.277 / 0.117	0.615 / 0.069	0.069 / 0.219	0.277 / 15.746	Beamposwork-around needed.
(p)-Ubuntu7MBP:	1.330	0.084	0.352	1.321	0.100	4.553	Bias ~ 0.74 ms.
(d)-TigerMBP	0.109	0.010	0.124	0.793	0.042	1.536	-
(d)-OpenML-ATI Linux-DRI2	0.036	0.002	0.026	0.494	0.023	0.614	Prototype
(p)-OpenML-Intel Linux-DRI2	0.705	0.046	0.169	0.920	0.063	0.699	Prototype

* Photodiode measurement procedure (p) adds approximately 0.5 ms bias and up to 0.2 ms variability.
* High precision timestamping provides considerable improvement over naive timestamping.
* Still not foolproof, e.g., graphics driver software bugs, extreme system overload. However, lots of builtin diagnostics.
* Doesn't work reliably for non-fullscreen displays and not at all for GUI's with desktop composition.
* Biased to report false deadline misses (false trial rejection) instead of false success in case of system overload.
* For a (hopefully) almost foolproof implementation: Watch out for GNU/Linux + DRI2 in 2011.

This work was supported by the Max Planck Society.

http://www.psychtoolbox.org

Random thoughts and comments, in no particular order...

- In case anybody wonders about this: The reason for the excellent performance of VISTAPC under medium load for Async flips without realtime scheduling is that realtime scheduling is partially enabled even for “non-realtime” Async flips under Windows Vista or Windows-7. Psychtoolbox enforces this partial realtime scheduling (MMCSS scheduling for the background flip thread).
- The configurations VISTAPC, LNXPC and RTLNXPC were all tested on the same Dell Optiplex-960 PC. This wasn't a standard off the shelf system, but an expensive turnkey system which is the host computer part of a commercial realtime eyetracker. One can assume that the PC hardware components were carefully selected by the engineers of the eyetracker vendor and that the Windows Vista system on that machine was carefully configured/optimized for good realtime behaviour. We aren't allowed to change anything in that systems configuration, not even graphics driver updates or software installation, otherwise our warranty and service contract would be void. One can't conclude from the good performance of Windows Vista that it would perform as well on any other hardware, but one can conclude that it is at least possible to get good performance from Vista if one invests significant effort into optimizing system setup and hardware selection.
- For the same reason one can't conclude that realtime Linux would perform equally perfect on any arbitrary hardware. There could be incompatibilities between hardware components or deficiencies in specific graphics device drivers that could ruin the performance even on a realtime operating system. The Linux setups weren't optimized in any way, but they were running on the same well configured PC hardware as the Vista system. One can conclude though that on suitable hardware, realtime Linux can achieve perfect single display graphics performance even under very high stress, sometimes without need for special configuration. Comparison of realtime Linux (RTLNXPC) with standard Linux (LNXPC) allows to separate how much of the performance (or lack thereof) can be caused by the hardware (and graphics drivers) compared to the robustness of the operating systems realtime scheduler and memory management. I always recommend installing realtime linux on a system. Under recent Ubuntu Linux distributions (e.g., 9.x, 10.x), this is easily achieved with the following steps:
    1. In a terminal window, type: sudo apt-get install linux-rt
    2. Enter administrator password to authorize installation of the realtime kernel.
    3. Wait for the installation to complete (less than 5 minutes).
    4. Reboot, select the realtime Linux kernel at the boot manager menu. You're done.
- Surprised by the poor performance of Windows XP I retested on a few other lab machines with different hardware (processor, chipset, graphics card) running WinXP and wasn't able to find one that performed better. Doesn't mean that WinXP is totally unsuitable for precise stimulus presentation on any hardware, just that it seems to be fairly easy to get it wrong – or difficult to find hardware where it works well.
- I was even more surprised by the poor performance of Windows-7, even with a totally idle 8-core high end system, high end graphics card, 12GB memory and up to date drivers, but so far couldn't find a setup that performed better. A much bigger problem of both Vista and Windows-7 is by design: Best performance in stimulus presentation quality and reliability can only be achieved if the system is using so called “page-flipping” to control stimulus onset. For some weird reasons, Microsoft designed their system so that only one fullscreen window in a session – the window with keyboard input focus and foreground activation status – can use pageflipping. There can only be one such window at any given time. This means that on a dual-display setup, stimulus presentation will become fragile. Suppose you have one monitor showing the stimulus presentation window for your subject, the other monitor showing the desktop and Matlab window for the experimenter. If the experimenter would perform any mouse-click on the desktop anywhere (or use alt-tab to switch keyboard input to the matlab window or other windows), the stimulation window would lose keyboard input focus and stimulus presentation accuracy would suffer greatly. If you try to do dual display stimulation (e.g., binocular presentation) you're out of luck as well, because only one of the stimulation windows could get page-flipping enabled, the other one would suffer timing glitches and visual artifacts, e.g., tearing, flicker and display corruption. The recommendation for people with need for high timing precision and multi-display setups is to stick to WinXP if you must, and long term consider/evaluate alternative operating systems, e.g., Linux.
  If you are stuck with Vista or Windows-7, you can configure Psychtoolbox for different tradeoffs wrt. Timing precision, performance:
    - Screen('Preference', 'ConserveVRAM', 2^18); at the top of your script allows to disable foreground activation. This will make the GetChar, CharAvail and ListenChar functions work again, but cause poor timing precision and possible visual artifacts in stimulus updating. By default, PTB tries to provide good timing on at least single display setups, but GetChar et al. Won't work at all on Vista and Windows-7 as soon as an onscreen window is open.
    - Screen('Preference', 'ConserveVRAM', 2^14); at the top of your script enforces use of the Windows DWM aka Aero desktop compositor instead of forcefully disabling desktop composition. The benefit of using DWM is that stimulus updates will be tear-free and free of other visual artifacts, even on dualdisplay setups – you can get nice looking animations. The downside is a complete loss of precision and control in terms of stimulus onset timing and timestamping. Stimuli won't show at the scheduled times, but an arbitrary, random (but usually small) number of video frames later, as the DWM has full control of when to present something on the screen and may decide to delay stimulus updates for reasons of efficiency or for pleasant visual updates (by Microsofts metric of pleasant and efficient). The returned presentation timestamps will be loosely related to true stimulus onset at best, but could be off by hundreds of milliseconds.

  After spending many days trying to improve the situation on Vista, Windows-7 (Psychtoolbox contains quite a few Vista/7 specific optimizations) I'm convinced that there isn't a good solution to these problems on either Vista or Windows-7. Maybe the situation will improve with Windows-8 at some time, maybe not.

A relatively new source of trouble that can interfere with presentation timing is the influence of the dynamic power management of modern graphics cards (GPU). Modern GPU's monitor their own average workload over a certain time window. If the load drops below a certain threshold, the GPU will reduce its core and memory clock speed and shut down parts of the chip and parts of the communication links (PCIe lanes) to the host computer to avoid wasting electrical power on unneeded performance. This reduces heat production and power costs and increases battery life on laptops. If average load increases over a certain threshold, the GPU will upclock again to a higher (more energy intense) performance state. The dynamic control algorithm has some inherent hysteresis and latency and is optimized for typical desktop use cases, not for the strict timing needs of vision scientists. The strategy of switching differs between GPU's and graphics driver versions. Timing problems for stimulus presentation (skipped deadlines) can arise if the GPU is idle or not busy enough for a period of time, then downclocks and then doesn't manage to upclock fast enough if there's sudden work to do. This is illustrated in the plot on the poster, where a simple permutation of trials (block 2) could suddenly cause many deadline misses, despite the fact that the same deadlines were easily satisfied for the regular schedule (block 1).

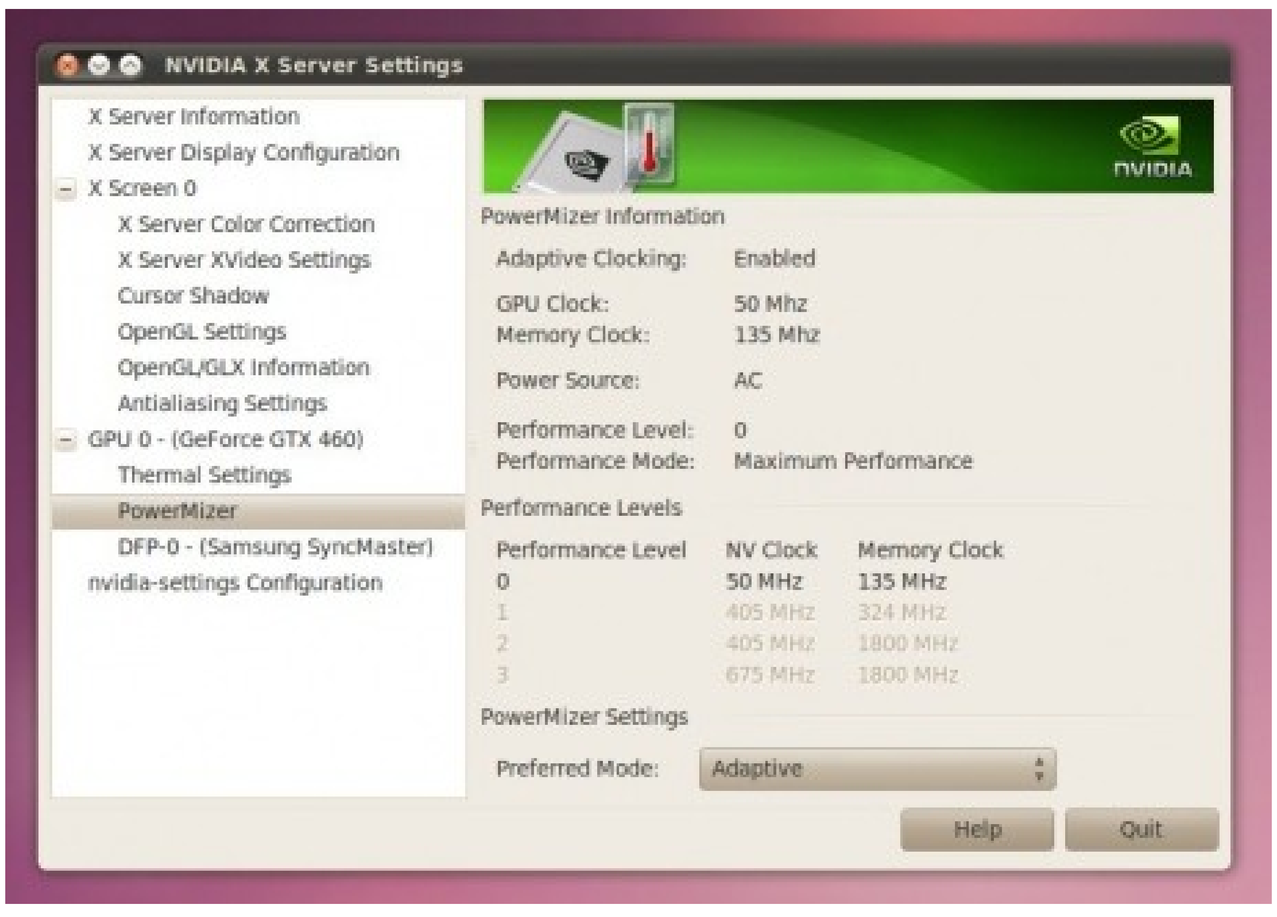
Random thoughts and comments, in no particular order...

When does this matter in practice?

- It doesn't matter for constant framerate animations where the stimuli in the animation frames are roughly equally complex. The GPU will quickly adjust to and stay at an appropriate performance level for your stimulus. It doesn't hurt to have some warmup trial though, so the system can prepare and adjust itself.
- It doesn't matter for stimulus presentation deadlines in the future if the deadlines 'twhen' are specified when using the Screen('Flip', win, when); command. Psychtoolbox will optimize internal processing of drawing commands to meet the specified deadline.
- It does matter if a stimulus is only presented in response to some external (unpredictable) event, e.g., a response by a subject (keypress, mouse press, eye movement detected by eye tracker, ...), or a trigger signal (fMRI, TMS, EEG etc.). An easy way to avoid problems is to first do all drawing in your trial code. Then call Screen('DrawingFinished', win, dontclear, 1); with 'dontclear' being the same parameter you'll pass to the Screen('flip',...,clearmode,...); command (typically simply [] for the default setting). Then do other stuff and then wait for the external event. The 'DrawingFinished' command will ensure that all pending rendering commands are finished as soon as possible and that your code only continues its execution after the stimulus image is completely ready for presentation. This way it doesn't matter if the GPU downclocks during the waiting period. At Screen('Flip', win); time, it will be easily capable of presenting your stimulus in time, even if it is downcloked to its lowest performance level.
- A problematic case is if you you have animations or stimulation sequences where none of the above applies and the complexity of your stimuli varies drastically from frame to frame. This would be the pattern simulated by the benchmarks conducted in block 2 of the presented benchmarks. The GPU may not be able to properly adjust itself and switch performance modes in the wrong moment, spoiling your presentation schedule. The only way to avoid this is to disable the GPU's dynamic power management and set the GPU to a fixed performance level sufficient for the most demanding stimuli in your experiment. The downside of the approach is additional work on your side and possibly a lot wasted electrical power, battery runtime and creation of excess heat and cooling noise on your computer. Unfortunately there doesn't exist a simple unified method of doing this, so I'll list some methods that are known to work:
  - As far as I know, on Apple MacOS/X there is no known way to disable dynamic power management, sorry.
  - On Linux and Windows with recent ATI GPU's and up to date drivers, you can use the command PsychGPUControl('SetGPUPerformance', gpuPerformance); with gpuPerformance set to a value between 1 and 10 to select constant performance (1 = lowest performance, max. energy saving, 10 = maximum performance, max power consumption) at the beginning of your script. At the end of your script you can call the function with a setting of 0 to reenable dynamic power management for normal desktop use.
  - On Linux with recent Nvidia GPU's and up to date drivers, there are two ways to select a fixed performance state: First via the control panel: Go to the PowerMizer section, set the PowerMizer preferred mode to “Maximum Performance” instead of “Adaptive”. See screenshot for guidance.
  - Another way to do it is to manually edit the xorg.conf file (typically inside /etc/X11/xorg.conf) and add add the following conf guration settings to the “Device” section of the X11 xorg.conf configuration file:

```
Option "Coolbits" "1"
Option "RegistryDwords" "PowerMizerEnable=0x1; PerfLevelSrc=0x2222; PowerMizerDefault=0x1; PowerMizerDefaultAC=0x1"
```

  - On WindowsXP, and for Windows Vista and 7 with Geforce GPU's older than Geforce-9000 series, the freely downloadable (for evaluation purpose) “Rivatuner” tool allows to tweak power management settings and to disable dynamic power management. You can google for the tool and follow the instructions in its manual. The procedure is non-trivial for beginners and requires manual modification of the Windows registry with all the risks involved.
  - On Windows Vista and Windows 7 with Geforce-9000 GPU's or more recent GPU's, the Nvidia control panel has a section “Power Management mode” that allows to set the power management to “Maximum Performance” instead of “Adaptive”.



Random thoughts and comments, in no particular order...

Precision / Reliability of stimulus onset timestamps:

- As can be seen by comparing the “Stddev” and “Range”columns between “high precision” and “raw” timestamping, PTB's high precision timestamping does provide significant improvements, especially when the system is under load. Worst case error (“Range”) for deviation between true stimulus onset time and reported time is below 1 msec in all cases, variance / jitter (Stddev) is usually in the low sub-millisecond range. These are the two parameters that usually matter for stimulus onset scheduling, checking for skipped frames, reaction time measurements and sync across modalities (audio video sync, sync with eeg, meg, tms etc.)
- The “MeanDiff” is the constant bias between measured onset time in the top-left corner of the display and the reported onset time (the 2nd return argument of [tvbl, tonset] = Screen('Flip', ...);) that is: MeanDiff = average over (tmeasured – tonset) for all trials and conditions. A perfectly working timestamping implementation should have a MeanDiff of zero. There are three reasons for different numbers:
  - Imperfections of the implementation and residual errors in the way the method works. This will typically introduce less than 100 microseconds of error.
  - Use of a photo-diode as a measurement device: The photo-diode itself, its placement on the monitor, phospor response times and low-pass filtering characteristics of the measurement electronics introduce a fixed bias of approx. 0.5 msecs and a variability of about 0.2 ms. For all measurements with the (p) label for photo-diode, the reader should subtract about 0.5 msecs from the reported values. Comparison with the measurements that employed the more accurate Datapixx device (d) show the difference/influence the photo-diode has and provide more realistic absolute values for “MeanDiff”.
  - Graphics driver bugs: Some graphics drivers for some GPU's on some systems have a small bug that causes reporting of timestamps that are about 0.74 msecs too early, thereby introducing a constant bias of 0.74 msecs, as noted in the “Caveat” column. The cause of this issue is well understood and the introduced error will always be constant and small (<< 1 msec). Psychtoolbox own implementation by now has a bugfix for this effect, so you'll always get bias free timestamps on Linux and on MacOS/X if you have an ATI GPU of the X1000, HD2000,HD3000,HD4000 series and use the PsychtoolboxKernelDriver (help PsychtoolboxKernelDriver for instructions). The problem may be present with certain drivers and GPU's from ATI, Nvidia or Intel if you use MacOS/X or MS-Windows.
  - Certain GPU's on MacOS/X and Windows have another class of graphics driver bugs (see label “Beamposworkaround” in the Caveat column). Psychtoolbox usually detects this class of bugs at startup and automatically enabled workarounds. If it doesn't detect the problem, the workaround can be manually enforced via adding the command:

    Screen('Preference','ConserveVRAM', 4096); If you need to enable multiple workarounds, add up all values to the 'conserveVRAM' setting. They all need to be applied in onle single call to the command.
  - The last rows (Linux-DRI2) denote preliminary results on a prototype implementation for the Linux free graphics driver stack (DRI2). As one can see, timestamping on ATI hardware is essentially perfect. If one discounts the error introduced by photo diode measurements in the OpenML-Intel case then timestamping on Intel hardware is also essentially perfect. We are working actively with the Linux graphics developer community to improve Linux's support for high precision graphics. The long term aim is to turn Linux into a solution for vision science applications that is far superior to existing offerings from Apple or Microsoft, but this is work in progress. First promising results of this work can be found for Intel GPU's in Ubuntu Linux 10.10 “Maverick” as released on 10.10.2010. More significant improvements are expected for the spring or summer 2011 editions of Linux distributions, e.g., Ubuntu 11.04 in April 2011.
  - Timestamping in general only works on fullscreen windows (and on Windows Vista or Windows-7 for exactly one fullscreen window on one display, as explained already). None of the existing operating systems allows any kind of reliable timestamping for non-fullscreen windows (e.g., as part of the regular GUI) or if desktop composition is enabled. For this reason, PTB disables desktop composition by default on Windows Vista/7 and MacOS. A simple way to achieve this under Linux is by typing this in a termina to disable composition for fullscreen windows:

    gconftool-2 -s --type bool /apps/compiz/general/screen0/options/unredirect_fullscreen_windows true

    Alternatively one can simply disable desktop composition to save even more resources. Future ptb versions will handle this automatically.

As a reference, the datasets and scripts that were used to perform the benchmarks for this poster can be found in the Psychtoolbox distribution inside the subfolder Psychtoolbox/PsychTests/ under the following names:

FlipTestConfigurations.zip, FlipTimingWithRTBoxPhotoDiodeTest.m: Script and input configuration files for performing the benchmarks.

PosterBatchAnalyzeTimestamps.m, BatchAnalyzeTiming.m, , AnalyzeTiming.m: Analysis scripts for analyzing the benchmark results created by FlipTimingWithRTBoxPhotoDiodeTest.m

You will need one of the supported measurement devices to collect external stimulus onset timestamps as “ground truth”: The UBW32/Bitwhacker + VideoSwitcher, or the RTBox with photo-diode, or the Vpixx DataPixx device. Alternatively you can run the script in no-measurement mode, in which case the absolute precision of the timestamps can't be evaluated.

You will also need some persistence, good nerves and luck, as the scripts and procedures are only sparsely (ahem) documented. Enjoy the beauty of research code...

This poster may get updated from time to time with new insights into the world of visual timing and timestamping.

History:

16.10.2010 – Initial version.