Behavior Research Methods
2007, 39 (3), 365-370

Adobe Flash as a medium for online experimentation: A test of reaction time measurement capabilities

Stian Reimers and Neil Stewart
University of Warwick, Coventry, England

Adobe Flash can be used to run complex psychological experiments over the Web. We examined the reliability of using Flash to measure reaction times (RTs) using a simple binary-choice task implemented both in Flash and in a Linux-based system known to record RTs with millisecond accuracy. Twenty-four participants were tested in the laboratory using both implementations; they also completed the Flash version on computers of their own choice outside the lab. RTs from the trials run on Flash outside the lab were approximately 20 msec slower than those from trials run on Flash in the lab, which in turn were approximately 10 msec slower than RTs from the trials run on the Linux-based system (baseline condition). RT *SD*s were similar in all conditions, suggesting that although Flash may overestimate RTs slightly, it does not appear to add significant noise to the data recorded.

Adobe Flash, formerly known as Macromedia Flash and colloquially called Flash, is an authoring application that allows programmers to create content for display over the World-Wide Web. Flash, which uses vector graphics and a combination of a programming language called Actionscript and simple frame-based animation, is most often used to create high-impact moving images to make attention-grabbing Web sites. Google indexes several tens of millions of Flash movies (the term *movie* is traditionally used to describe Flash-created content, even though, as we shall see, it captures only one of Flash's capabilities), and Flash is used especially for creating animations, games, splash screens, and, increasingly, advertisements. As such, Flash provides one of the key means of supplying interactive content to Web users.

In the past, Flash was used predominantly for animation. However, the development of Actionscript has meant that, in principle, Flash can be used to implement psychological experiments of almost any design, subject obviously to the constraints of Web-based experimentation. Psychologists have begun to use Flash for running surveys and experiments over the Web, often replicating results obtained in the laboratory (see, e.g., Reimers, 2007; Reimers & Maylor, 2005, 2006).

Although there are many different ways of running experiments over the Web—the dominant method involves using HTML forms to conduct surveys—a key advantage of Flash is that it allows the accurate measurement of time, which could make it suitable for research based on reaction time (RT). Flash's suitability for psychological experimentation has been investigated only in limited fashion. Schmidt (2001) compared presentation duration accuracy using several Web animation methods. He found that Flash performed adequately on faster computers but deteriorated significantly on slower machines. Since the slower machines ran at 60–75 MHz and the faster machines ran at only 400–500 MHz—meaning that both types are obsolete now—it is unclear how modern versions of Flash Player, run on more powerful computers, would perform on a similar test.

We know of no studies in which Flash's accuracy in measuring RTs has been quantified. Since Flash runs within a browser and has no control over interruptions from other programs, there are reasons to suspect that Flash cannot record RTs as accurately as other experiment implementation methods. Our aim in the present experiment was to investigate the degree of accuracy with which Flash measures RTs relative to a baseline system that has a demonstrated accuracy to the millisecond. We also aimed to compare the performance of Flash on controlled laboratory computers with its performance on computers of participants' choosing outside the laboratory over which we had no control.

Introduction to Flash Player

Setting up and running Flash-based experiments involves two components: an authoring program, which the psychologist/programmer uses to develop and export experiments in Shockwave Flash (.swf) format, and the application player, which participants use on their own computers to run .swf files. Confusingly, both of these components are often referred to simply as Flash. Here, we adopt the convention of calling the authoring program *Flash* and the application player *Flash Player*. Addition-

S. Reimers, s.reimers@ucl.ac.uk

ally, although we discuss only Adobe's proprietary authoring tool and application player, other software for creating and playing Flash files exists, including the authoring program SwishMax and the open-source Flash player Gnash.

Flash Player is generally run as a Web browser plug-in, and it is installed on the majority of Internet-enabled computers: In March 2007, Adobe reported that Flash Player was installed on 98.7% of Internet-enabled desktop computers in the mature markets of the U.S., Canada, U.K., Germany, France, and Japan (Adobe, 2007). Flash Player has been developed for several different platforms, including Windows, Linux, and Mac OS, as well as for operating systems used in handheld devices, including Palm OS and Symbian.

Like Java, Flash Player runs as a sandboxed virtual machine. Data can be stored on the client machine and accessed later (in a manner similar to that used by cookies), but otherwise, access to the client machine's hard drive is not permitted. Data can be transmitted from Flash Player across the Web to a remote server but only to the domain at which the Flash movie was hosted. In other words, a Flash experiment accessed at www.sitea.ac.uk/Experiment1.swf could pass data to www.sitea.ac.uk/cgi-bin/datahandler.pl but not to www.siteb.ac.uk/cgi-bin/datahandler.pl. Data can be sent using the standard GET or POST methods, which are used to send HTML forms.

Initializing the player is fast and automatic, meaning, in principle, that users can be unaware that the plug-in has started running. However, in April 2006, an update to Internet Explorer (IE) changed the way that plug-ins initialized, which meant that IE users had to click on an embedded Flash movie to interact with it (recently, Javascript-based techniques have been developed to circumvent this problem). Flash Player is also small in size and does not make heavy demands on a system in order to run; recommended minimum system requirements for Windows PCs and Macs are a 500-MHz processor (Pentium II or G3) and 128 MB of memory.

Introduction to the Authoring Program Flash

The authoring program Flash uses a combination of frame-based animation and Actionscript code associated with frames and objects. The frames work as they do in a traditional movie. The speed of the movie is set globally in frames per second (fps), and the contents of each frame are displayed in sequence at the specified frame rate. Flash contains an interface for drawing items within a frame, using standard graphics tools such as paintbrush, pencil, line, and circle. These items can be defined as objects, which can then be addressed by Actionscript code, which can be used to control the object's attributes.

Animation involves copying and changing objects across frames. For example, one could have a red square object at coordinates (50, 50) in Frame 1, copy it to Frame 100, and move it to coordinates (450, 450). By selecting `create motion tween`, the position of the square in frames 2–99 is interpolated. Thus, at a playback speed of 50 fps, when playing frames 1–100, the square would move smoothly across the screen in 2 sec. Similar animation techniques can be used on a number of attributes an object may have, such as magnification, position, rotation, transparency, brightness, and skew.

Although animation techniques allow the creation of visually pleasing movies, it is the Actionscript code that allows full experiments to be generated. Actionscript can be associated with frames and objects. Frame-based Actionscript is executed unconditionally when a frame is entered and can include a `stop();` command, which prevents the movie from advancing to the next frame until a `play();` or similar command is given. Object-based Actionscript code uses conditional statements that refer to user input. Often, the conditional statements refer to manipulation of the object itself. For example, the statement `on(press){x+=1;}` would increase the value of `x` incrementally each time the participant clicked on the object in question. However, the conditional statements can also refer to keyboard input: For example, `on(keyPress"<Space>") {x+=1;}` would increase the value of `x` incrementally each time the participant pressed the space bar.

The combination of frame-based animation and Actionscript code within a frame or object can be confusing to a programmer. However, this combination does allow for the integration of eye-catching, user-friendly animations, which can be used to lead a participant through an example of the task they are about to perform, with the flexibility to implement complex experimental designs.

Suitability of Flash for Psychological Testing

Clearly, all methods for performing tests using the Web have advantages and disadvantages. The similarity in architecture between Flash and Java (both are client side, sandboxed, and run as virtual machines) means that the two share many of the same advantages and disadvantages. Briefly, their advantages over JavaScript, for example, include more precise control over stimulus appearance and duration and more accurate timing. Advantages over specialist packages such as Authorware include having a more ubiquitous plug-in, which means that most participants are not forced to visit a third-party Web site and download a plug-in before they can participate. Disadvantages of Flash and Java include the relative complexity of their programming languages and possible differences in performance across platforms.

The main advantages Flash has over Java include the ubiquity of its plug-in, the shorter initialization time (Java can take several seconds to start up), and the familiarity Web users have with Flash-based content. Additional compelling advantages are the graphical user interface and frame-based structure of Flash, which make it easy for a relatively inexperienced programmer to create ergonomic content, particularly when using preprepared components like drop-down menus, text boxes, and buttons. One advantage of Java is that it is a more established and rigorous programming language, although Flash is improving along those lines. Also, Java code can be adapted easily for use in other domains, such as nonnetworked computers and handheld devices. Again, Flash has recently begun to

develop in these areas, and basic versions of Flash Player are beginning to be found on mobile devices. We have recently implemented simple experiments on cell phones using both Flash and Java, thus demonstrating the potential of both languages to run on other devices.

Flash has enough potential advantages as a testing medium for it to be worthy of consideration. Its disadvantage is that, as a new medium for psychological experimentation, it has not been thoroughly tested. In particular, one key factor that appears largely untested is the accuracy with which Flash can measure RTs. There are few options for Web experimenters wishing to measure RTs with better than 100-msec accuracy. Authorware and Java are perhaps the only current alternatives that allow relatively accurate measurement of RTs over the Web.

In this experiment, we consider two types of error that can distort RT measurements: random and systematic. Random error can come from quantizing errors and variability in timings of stimulus onset to response. A combination of random and systematic error can come from the delays in sampling the keyboard, interruptions from other programs that delay immediate detection of key or mouse presses, and delays such as the delay between initiating a procedure to put an image on the screen and the image's actual appearance. Systematic error may cause recorded RTs to be substantially different in mean, but not *SD*, from those recorded perfectly (e.g., RTs recorded over the Web may all be 50 msec slower). Conversely, random error should cause recorded RTs to be substantially different in *SD*, but not mean, from those recorded perfectly.

The two types of error have different implications for interpretation of noisy data. Systematic error makes it difficult to compare means of data from experiments using different measurement apparatuses. However, it is rare to find direct numerical comparisons among different experiments, since experiments generally compare the effects of an experimental manipulation between or within subjects, using a single measurement procedure. Although random error does not affect the means of any results, the increase in variance may lead to an experiment's losing power to detect small differences among group means. That said, the addition of random error on the order of tens of milliseconds often makes very little difference to tests of differences between groups (Ulrich & Giray, 1989). Additionally, if the amount of error is relatively small, it can be compensated for by increasing the number of trials (so that the estimate of participant means is more accurate) or the number of participants (so that the estimate of group means is more accurate) or both. However, there are practical limitations on the extent to which the number of trials or participants can be increased; thus, it is worth investigating the amount and types of errors that Flash adds to any RT measurements in order to evaluate whether Flash is a viable medium for conducting online experiments.

The Present Study

The present study was designed to allow us to compare RTs measured by Flash with those measured by a system known to be reliable, under experimental conditions. Our reliable baseline condition, which is known to measure RTs with millisecond accuracy, was programmed in C and used the same setup described in Stewart (2006a, 2006b). We compared this condition with two Flash conditions. In the lab Flash condition, the setting and procedure were identical to those of the baseline condition. Participants were seated at the same computers and were given the same instructions. In the nonlab Flash condition, participants were asked to find a Web-connected computer and complete the Flash experiment on their own time. The task we used was choice reaction time (CRT). We implemented an experiment measuring CRT once in Flash and once in the baseline condition programmed in C, keeping the appearance of the experiment as similar as possible across implementations.

METHOD

Participants

Twenty-four undergraduate psychology students at the University of Warwick participated in the experiment for course credit.

Design

There were three within-subjects conditions: baseline, lab Flash, and nonlab Flash. Order of conditions was counterbalanced, subject to the constraint that baseline and lab conditions occurred consecutively (to avoid having participants make two visits to the lab), thus creating four orders: (1) baseline, lab, nonlab; (2) lab, baseline, nonlab; (3) nonlab, lab, baseline; and (4) nonlab, baseline, lab. In all conditions, participants classified rectangles that appeared on the screen as red or green by pressing a key on a buttonbox (baseline) or keyboard (lab Flash and nonlab Flash). The experiment comprised 30 such binary classifications. All trials were independent: On each trial, the chance that the rectangle would be red or green was equal (.5). Response key mapping was counterbalanced: Half of participants pressed the left key for a green rectangle, and the other half pressed it for a red rectangle. Participants pressed the same response key (left or right) in all three conditions.

Implementation

Flash (lab and nonlab conditions). A 750 × 500 pixel Flash movie, coded using Flash MX, was embedded in a gray Web page. The first frame of the movie requested the participant's ID number for counterbalancing purposes. The second frame contained instructions. The third contained the Actionscript code that ran the experiment. An invisible movie clip was embedded in the movie frame, containing an `onClipEvent(keyDown)` event handler. The procedure for measuring RT was as follows. In line 1, a call was made to make either the red or green rectangle objects visible, setting `MovieClip._visible = true`. In line 2, the number of milliseconds that had elapsed since the movie started was recorded, using the `getTimer()` function. (Whether the timer was polled before or after the command to make the rectangle visible made no difference to the RT measurements.) In line 3, the flag `allowresponse = true` was set, mean-

ing the event handler would accept keypress responses. When a key was pressed, the event handler checked that `allowresponse == true`. If so, it recorded the number of milliseconds that had elapsed since the movie started and subtracted the value recorded at the previous poll, when the target was made visible, thus giving the RT. It then reset `allowresponse = false`, meaning that further keypresses would be ignored, and initiated the display of feedback. At the end of the experiment, a string of RTs and accuracy was sent to a perl script on a remote server using the POST method; the perl script saved the data to a text file.

C (baseline condition). A C program using the X Window System to display stimuli (Stewart, 2006a) and a parallel port button box (Stewart, 2006b) was used. The program was scheduled as `SCHED_FIFO` using the `sched_setscheduler()` system call to prevent interruption from other tasks running on the system. The memory used by the program was locked using a `mlockall()` system call to prevent the memory from being swapped to disk. RTs were measured using the `gettimeofday()` system call, which provides microsecond accuracy. Finney (2001) discussed these techniques in detail.

RTs were measured from the end of the vertical retrace upon which the stimulus was drawn to the first detection of a close of a button on the parallel port button box. The error in this measurement was determined by repeatedly sampling the parallel port and then measuring the difference between the last time that the button had not been pressed and the first time that it was pressed. In all cases, the error was less than 1 μsec.

Procedure

The system used for the lab experiments (baseline and lab Flash) was a dual-processor 1.4-MHz AMD Athlon with 256 MB of RAM with a PCI NVidia GeForce 2 MX with 32 MB of video RAM. We used the Debian Sarge GNU distribution with a 2.4.27-2-k7-smp Linux kernel, XFree86 4.3.0, and the NVidia 1.0-5336 driver4 (not the NV driver that comes with XFree86). The CRT monitor was a Sony CPD-G220 Color Trinitron running with a refresh rate of 85 Hz. The C code was executed from the command line; the Flash program was run by typing the URL of the testing page into a Firefox Web browser. For the nonlab condition, participants used their own computers or student computers at the University.

The experiment began with instructions presented on a full-screen, borderless gray window. Each trial began with a "+" prompt in the center of the screen. After a random, uniformly distributed interval of between 1,500 and 3,000 msec,[1] a red or green 200 × 100 pixel rectangle was displayed in the center of the screen until the participant responded. After the response, either "correct" or "wrong" appeared on the screen immediately as feedback. The feedback was followed by a 2,000-msec intertrial interval during which the screen was blank; then the next trial began. At the end of the experiment, data were written either to a remote server (in lab Flash and nonlab Flash conditions) or to the local hard disk (in the baseline condition).

RESULTS

One participant's data were discarded because the participant was assigned the wrong ID number. To allow appropriate counterbalancing, data from the participant who immediately followed the excluded participant were also excluded. This left 22 participants' data in the analysis.

There were two main aims of the analysis. The first was to investigate, by comparing group means and *SD*s in each condition, the extent to which the statistics typically reported in an experiment's results sections are affected by the experiment's location and implementation. The second was to examine the data more qualitatively, to determine whether RTs in all conditions followed the same distribution shape or whether in Flash they were quantized or skewed in ways that could have affected the interpretation of results.

The group means and *SD*s, along with error rates, are given in Table 1. Only RTs of less than 1 sec (100% of baseline trials, 99.5% of lab Flash trials, and 99.2% of nonlab Flash trials) were included in the analysis. One baseline trial of 0 msec and one nonlab Flash trial of 13 msec were excluded. All other RTs were in the range of 203–956 msec.

To test the significance of any differences in RTs and *SD*s, we constructed a general linear model with between-subjects variables of in-lab counterbalancing (baseline then Flash, Flash then baseline) and lab Flash/nonlab Flash counterbalancing (lab Flash then nonlab Flash, nonlab Flash then lab Flash) and within-subjects variables of condition (baseline, lab Flash, and nonlab Flash). With participants' RT means as the dependent measure, there was a main effect of condition [$F(2,36) = 4.07, p = .026$], and no other main effects or interactions. A similar effect was found with medians as the dependent measure [$F(2,36) = 6.18, p = .005$]. With participants' RT *SD*s as the dependent measure, there were no significant effects or interactions (all ps $> .100$). With interquartile range as the dependent measure, the only significant effect was in the lab Flash/nonlab Flash counterbalancing variable

Table 1
Differences in Averages and Variabilities of Reaction Times (RTs, in Milliseconds) Across Testing Conditions and Error Rates

Condition	Mean of RT Means	Mean of RT *SD*s	Mean of RT Medians	Mean Interquartile Range	Mean Range	Error Rate (%)
Baseline	376.5	89.2	363.6	101.0	380.8	4.0
Lab	386.6	83.0	376.8	96.2	369.0	4.4
Nonlab	407.9	87.4	402.5	101.8	379.5	5.6

[$F(1,18) = 6.36$, $p = .02$]. Finally, using range as the dependent variable, there was a significant effect only in the interaction between the two counterbalancing variables [$F(1,18) = 4.93$, $p = .04$].

Estimates of the average and the spread of RTs in the different conditions suggest that RTs recorded with Flash were 10–40 msec longer than those recorded in the baseline condition. The second aim now was to look qualitatively at the individual RTs, to investigate whether there were any effects of quantizing or any large differences in the distribution of RTs. To do this, we constructed a cumulative frequency distribution (CFD) chart for the three conditions (Figure 1). Since we were primarily interested in any mechanical differences among the conditions, we included every datapoint from all 22 participants in the analysis (including the short and long RTs excluded from the previous analysis but still excluding the single 0-msec RT from the baseline condition). Any quantizing would be detectable as step-like patterns in the CFD, and any differences in the shapes of the RT distributions would be detectable because the CFDs would deviate from *x*-axis-shifted copies of each other.

There was no evidence of quantizing, suggesting that in all three conditions, RTs were not constrained to a limited set of values (although this result does not rule out quantizing elsewhere in the system). There was, arguably, a small difference in the distribution of RTs between the lab Flash condition and the other two conditions, as shown in apparent gradient differences of the CFD. This is clearly a small effect, and although further research into it is merited, a bias of this magnitude does not appear to undermine the viability of Flash in general, particularly since results from the more realistic nonlab Flash condition did not appear to be different in shape than those from the baseline condition. However, this effect does suggest that the distribution of RTs may be slightly different in the lab Flash condition than in the two other conditions.

DISCUSSION

The comparison of RTs measured using a baseline condition accurate to the millisecond versus Flash run on laboratory computers versus Flash run on Web-connected computers outside the lab suggests that there are small differences in the measured RTs across the conditions. Compared with the mean RTs from the baseline condition, RT means for the lab Flash condition were around 10 msec longer, and RTs for the self-administered, nonlab Flash condition were 30–40 msec longer. There were no significant differences in the RT *SD*s across the three conditions.

We set out to investigate whether conducting experiments using Flash would add random and/or systematic error to RT data. Our conclusion, drawn from the results above, is that Flash adds a small amount of systematic error to RTs, in that the RTs measured in the lab Flash condition were longer than those measured in the baseline condition. Small errors in RT measurement are largely unavoidable. Given that a screen refresh takes 10–20 msec, any RTs based on display of visual stimuli are likely to have a certain amount of systematic error added to them, unless the issue of screen-refresh time is explicitly dealt with (as it was in the baseline condition).[2]

Flash does not appear to add significant random error to RT measurements. Ignoring conspicuously long RTs

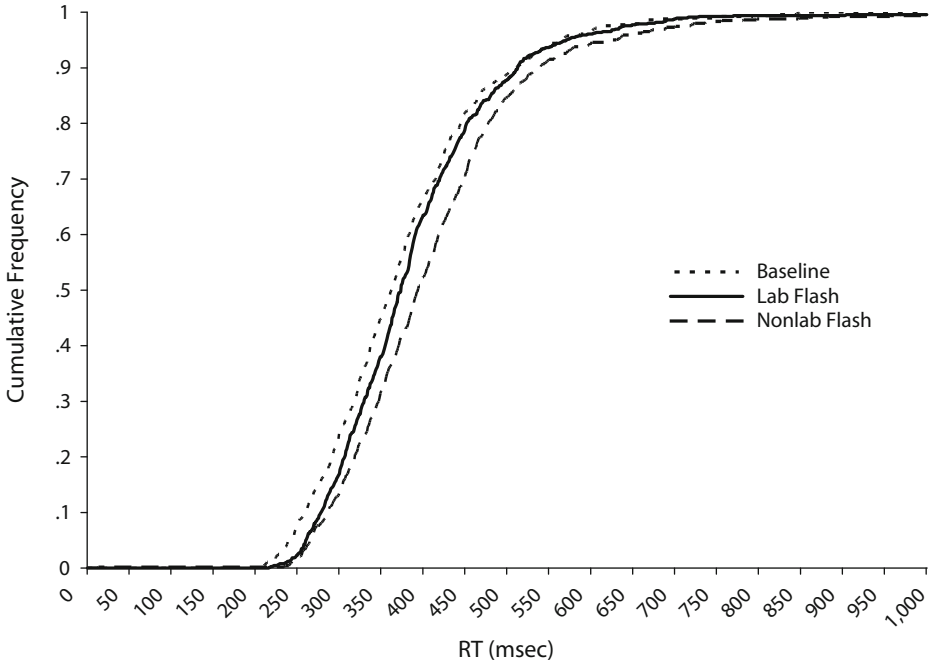

Figure 1. Cumulative frequency of trials as a function of reaction time (RT, in milliseconds) for the baseline (accurate to the millisecond), laboratory-administered Flash, and nonlaboratory Flash conditions.

(>1 sec), *SD*, interquartile range, and full range were the same in all three conditions. Unlike in the baseline condition, in the Flash condition, a small number of RTs were longer than 1 sec (although only in approximately .5% of responses). Some of these longer RTs in the nonlab Flash condition may have been due to distractions, but others may have been due to interruptions from other active applications on the computer. Whatever the cause, these data suggest that searching for and eliminating outliers is particularly important when looking at Flash data.

Use of Flash in the Lab and Over the Web

A final finding of interest is that RTs from the nonlab Flash condition were, on average, longer than those from the lab Flash condition, even though both used Flash. There is a confound here between physical and psychological differences between the conditions. In the lab condition, Flash was run under Linux, and no other user applications were active (apart from the Web browser in which Flash was displayed). In the nonlab condition, it is likely that the majority of participants used networked Windows machines. It is also likely that they had other applications open as they completed the experiment (e.g., most received an e-mail with a link to the testing site, meaning that their mail program was probably also active). Similarly, there are clear psychological differences between participating in a laboratory experiment (where an experimenter is present, the time of participation is fixed, and the room is quiet and contains no distractions) and at one's own computer. In the experiments run on Flash outside the lab, we wanted conditions to be as similar as possible to those encountered by Web surfers arriving at a site; we therefore did not instruct participants to close other applications or to minimize distractions. Thus, it is increases our confidence in Flash that there was no more random error in the nonlab Flash condition than in the baseline condition, and that mean RTs were only 30–40 msec longer than those from the baseline condition. Of course, this was not a perfect simulation of the conditions under which participants generally complete an experiment on the Web. Our participants had met, or were going to meet, the experimenter, they knew that they had to complete the task for course credit, and half had completed the task already in the lab. In addition, the range of computer types used by our participants was probably smaller than those used by participants in Web-based research. Thus, we do not want to generalize too strongly from these results to all potential Web-based Flash studies.

Similarly, software and hardware evolve at a rapid rate. Since we drafted this manuscript, an updated authoring tool, Flash CS3, has been released, which includes support for Actionscript 3.0. This will change the way in which experiments can be coded. New versions of Flash Player have also been released, 8 and 9, that have been designed to improve performance. It is therefore likely that RT accuracy will change slightly as the authoring program and application player develop. However, the results presented here suggest that, in principle, Adobe Flash is a viable method for running large-scale RT-based experiments.

AUTHOR NOTE

We are grateful to Petko Kusev for his assistance in running the experiment. This research was supported by a grant from HSBC Bank to S.R. Correspondence concerning this article should be addressed to S. Reimers, who is now with the Department of Psychology, University College London, 26 Bedford Way, London WC1H 0AP, England (e-mail: s.reimers@ucl.ac.uk).

REFERENCES

ADOBE FLASH PLAYER STATISTICS (2007). Retrieved August 16, 2007, from www.adobe.com/products/player_census/flashplayer.

FINNEY, S. A. (2001). Real-time data collection in Linux: A case study. *Behavior Research Methods, Instruments, & Computers*, **33**, 167-173.

REIMERS, S. (2007). The BBC Internet study: General methodology. *Archives of Sexual Behavior*, **36**, 147-161.

REIMERS, S., & MAYLOR, E. A. (2005). Task switching across the life span: Effects of age on general and specific switch costs. *Developmental Psychology*, **41**, 661-671.

REIMERS, S., & MAYLOR, E. A. (2006). Gender effects on reaction time variability and trial-to-trial performance: Reply to Deary and Der (2005). *Aging, Neuropsychology, & Cognition*, **13**, 479-489.

SCHMIDT, W. C. (2001). Presentation accuracy of Web animation methods. *Behavior Research Methods, Instruments, & Computers*, **33**, 187-200.

STEWART, N. (2006a). Millisecond accuracy video display using OpenGL under Linux. *Behavior Research Methods*, **38**, 142-145.

STEWART, N. (2006b). A PC parallel port button box provides millisecond response time accuracy under Linux. *Behavior Research Methods*, **38**, 170-173.

ULRICH, R., & GIRAY, M. (1989). Measuring reaction times: How accurate must a clock be? Good news for bad clocks! *British Journal of Mathematical & Statistical Psychology*, **42**, 1-12.

NOTES

1. In the Flash implementation, the movie frame rate was set to the default 12 fps. This had the unforeseen effect that the check as to whether the interval duration had been reached was also made only 12 times per second, even though the experiment was contained within a single frame. Thus, in Flash, the intervals that participants experienced were quantized and on average 42 msec longer than those in the baseline condition. It is unlikely that this made any difference in performance, given the small (i.e., 2%) difference in mean delays, the sub-100-msec quantizing, and the noise added to the quantized intervals by the monitor refresh rate. More importantly, RT measurement was unaffected by frame rate and thus was not quantized.

2. Although a screen refresh rate of 50 Hz would be expected to add 10 msec of systematic error to RTs (assuming it added uniformly distributed noise with a range of 20 msec), the random error it contributed would be negligible. This is because the new *SD* would approximate the square root of the sum of the squares of the participant's *SD* and the noise *SD*. In a simulation, we found that the effect of a 50-Hz refresh rate was to add only approximately 0.2 msec to RT *SD*s.

(Manuscript received January 9, 2006;
revision accepted for publication March 3, 2006.)