Behavior Research Methods
2010, 42 (4), 1096-1104
doi:10.3758/BRM.42.4.1096

PsyToolkit: A software package for programming psychological experiments using Linux

Gijsbert Stoet
University of Leeds, Leeds, England

PsyToolkit is a set of software tools for programming psychological experiments on Linux computers. Given that PsyToolkit is freely available under the Gnu Public License, open source, and designed such that it can easily be modified and extended for individual needs, it is suitable not only for technically oriented Linux users, but also for students, researchers on small budgets, and universities in developing countries. The software includes a high-level scripting language, a library for the programming language C, and a questionnaire presenter. The software easily integrates with other open source tools, such as the statistical software package R. PsyToolkit is designed to work with external hardware (including IoLab and Cedrus response keyboards and two common digital input/output boards) and to support millisecond timing precision. Four in-depth examples explain the basic functionality of PsyToolkit. Example 1 demonstrates a stimulus–response compatibility experiment. Example 2 demonstrates a novel mouse-controlled visual search experiment. Example 3 shows how to control light emitting diodes using PsyToolkit, and Example 4 shows how to build a light-detection sensor. The last two examples explain the electronic hardware setup such that they can even be used with other software packages.

Currently, there are various tools for developing software capable of running psychological experiments. Well-known commercial software packages are Superlab, E-Prime, Inquisit, and DirectRT (Stahl, 2006). These software packages take care of timing precision and support specialist hardware, such as keyboards designed for reaction time measurement. PsyToolkit has similar functions but differs from the commercial packages in that it is developed specifically for the operating system Linux.

Linux's popularity is steadily increasing, in part because most of its software, including PsyToolkit[1], is free of cost. This makes Linux and PsyToolkit suitable for those on a limited budget, such as students or universities in developing countries. Furthermore, with the addition of PsyToolkit, Linux is well suited for use by experimental psychologists. After all, statistical software for data analysis and office software for dissemination and communication have been around for Linux for a long time, and a fully functional experimental toolkit was probably the only missing bit.

Although PsyToolkit is the only experimental package developed *exclusively* for Linux, it is not the only software toolkit available for programming experiments on Linux computers. Today, there are a number of cross-platform solutions that have been reviewed elsewhere, such as PyEPL (Geller, Schleifer, Sederberg, Jacobs, & Kahana, 2007), PsychoPy (Peirce, 2007), Vision Egg (Straw, 2008), and TScope (Stevens, Lammertyn, Verbruggen, & Vandierendonck, 2006). Cross-platform solutions are ideal in principle, because they aim to deliver the same software package for a number of different operating systems—typically, Microsoft Windows, Apple Macintosh, and Linux. But in practice, cross-platform solutions rarely deliver exactly the same functions on all three platforms, and it is typically the Linux implementation that suffers from a lack of complete implementation.

PsyToolkit has three components, which can be used independently. From a user perspective, the main component of PsyToolkit is its scripting language, designed for psychological experiments and distributed with a number of detailed examples, including a spatial stimulus–response compatibility experiment, a task-switching experiment, an inhibition-of-return experiment, a computer analogue of the Wisconsin card-sorting test, a Fitts's law experiment, and a visual search experiment. Many of the examples also include R scripts for data analysis with the free and open source statistical software R (R Development Core Team, 2009).

In the scripting language, experiments are programmed as a list of command codes (similar to most programming languages). There is no graphical user interface to program experiments, as is available in various other psychological software packages; the list of command codes must therefore be entered and edited using a text editor. The scripting language is optimized for psychological experiments, with command codes for events of specific relevance to experiments. The efficiency of the scripting language is illustrated in Appendix A, which shows an example of a fully functional stimulus–response compatibility experiment in as few as 31 lines (the examples will be explained in detail below).

G. Stoet, g.stoet@leeds.ac.uk

After a script has been created, the user can use the program "psycc" to compile it into Linux executable code (the user types "psycc" on the command line interface, followed by the name of the script). "Psycc" is a source-to-source cross compiler; that is, it translates the script code into C code, which then is compiled into machine code (i.e., the code the microprocessor uses). In accordance with the Linux tradition, "psycc" comes with a manual page (which can be shown typing "man psycc") and multiple options. For example, if the user wishes to get an overview of the scripting language syntax, the user types "psycc -s," and the syntax will be displayed in the terminal window.

A script is a list of command codes, and this list follows a syntax. At the highest level of the scripting language, there are different sections. The "options" section can be used to set specific options the user might have for the experiment. Currently, there are 19 different options. For example, the "cedrus" option enables the use of Cedrus keyboards, and the "resolution" option can set another screen resolution than the default 800 × 600 pixels. Sections are separated from each other by an empty line, and indentation of code is recommended (but not required) for readability (see the Appendixes for examples).

Apart from the "options" section, there are seven other sections available. Four of these sections are used just to specify stimulus material ("bitmaps," "sounds," "videos," "fonts"). If an experiment uses none of these stimulus types, these sections are not necessary (see Example 1 below for a simple experiment that can do without them). Further sections are for describing the events of individual trials ("task"), describing different experimental conditions ("table"), and describing the way trials are selected and combined ("block").

The "task" and "block" sections allow for a high level of control and design complexity. For example, users can use variables, control structures (i.e., "if . . . then"), and store data (i.e., "save"). The "task" section, especially, is designed to present stimuli (e.g., "show bitmap") and process signals from the keyboard(s) and mouse. Altogether, 30 different command codes can be used in the "task" section, most of them with various parameters. Different task sections can be used to describe different tasks or task types used in an experiment. For example, if you wanted to combine a visual search and a stimulus–response compatibility experiment in one study, you would have two different task sections.

The "block" section (with 11 different command codes) is designed to specify how trials are combined into blocks. The user can sequence trials in many different ways. For example, the user can randomize the call of trials or use a fixed order of trials (which can be used to ensure that all participants perform exactly the same order of trials). Furthermore, trials can be called such that they will repeat when the participant makes an error. Also, the maximum duration of a block can be specified with one command ("maxtime"), which is helpful for experiments in which the researcher wants to limit the time a participant spends on a set of trials. Finally, the block statement has a number of command codes available for inserting breaks in the experiment, displaying instructions, setting variables, and even calling other programs (e.g., the statistical programming language R to analyze the participant's data and present the participant with performance feedback).

The second component of PsyToolkit is a C library, which offers the same functionality as the scripting language, but which can also be used separately. Nearly all commands used in the scripting language have an equivalent in the C library.[2] A quick way to learn the C library is to compile a script with "psycc" and use the "−k" option, which keeps a copy of the temporary C file into which the script has been compiled. Users who want features not easily performed by the scripting language can start with a script, keep the C code, and then enhance the C code. But users who want to add just a few lines of C can do this in a much more convenient way by using the command code "c" (whereby everything following "c" will be interpreted and executed as C language).

The third component of PsyToolkit is a questionnaire presenter ("PsyQuest") that enables a quick and easy entry system for questionnaires using the mouse or keyboard. The functionality and implementation of "PsyQuest" is entirely different from those of the previous two components of PsyToolkit. It is included to enable researchers to complement their experiment by collecting demographic data or carrying out established personality tests.

So far, this review has focused on the programming interface, but this interface would not be of much use without some of the underlying programming technologies that ensure accurate timing and access to hardware, such as the screen, response pads, and digital input/output boards.

Timing accuracy in multitasking operating systems has received much attention in the psychological research methods literature, and accurate timing is not a problem for Linux-operated computers (Finney, 2001). But even when the Linux system is capable of accurate timing, external devices such as screens, keyboards, and computer mice might affect timing because of their own timing inaccuracies.

Both traditional cathode ray tube (CRT) screens and modern flat screens (LCD) are typically very reliable in their timing and similar across product lines, and the use of vertical synchronization, as recommended for psychological experiments on all platforms (Gofen & Mackeben, 1997; McKinney, MacCormac, & Welsh-Bohmer, 1999; Stewart, 2006), is implemented in PsyToolkit. It is possible to measure the timing accuracy of visual stimulus onset and offset using the BlackBox toolkit (Plant, Hammond, & Turner, 2004; Plant, Hammond, & Whitehouse, 2002) or with a setup similar to that in Example 4 (below). More difficult to deal with are timing inaccuracies that can arise from response measurement with the regular keyboard and the mouse (Plant, Hammond, & Whitehouse, 2003). These devices vary strongly in their timing accuracy and should be tested in the lab if possible—for example, using the BlackBox toolkit. For time-sensitive tasks, PsyToolkit should use an external keyboard, such as IoLab[3] or Cedrus (PsyToolkit has a number of dedicated procedures for dealing with these keyboards). These devices keep their

own time, independently of the host computer. For less time-sensitive tasks, the regular keyboard and mouse can be extremely helpful and cost-effective solutions.

For users who want to construct their own devices for stimulus presentation or data collection, PsyToolkit supports two digital input/output boards (i.e., a computer card that can send or receive discrete on/off signals to or from an external electronic circuit). On the one hand, it supports the parallel port, which is often built into desktop PCs, and on the other hand, it supports Measurement Computings' PCI-DIO24 board. Example applications for an input/output board are a stimulus display made with light emitting diodes (LEDs, see Example 3), which can be updated with a millisecond resolution (unlike regular computer displays, which are typically updated in the 60-Hz range), and a light sensor system (see Example 4).

Like any other software package, PsyToolkit keeps increasing its functionality with each version. The potential downside of continuing development is the difficulty of between-version compatibility. To deal with this, PsyToolkit has been designed and implemented such that multiple versions of PsyToolkit can be installed on one computer, and can be used independently of each other. Laboratories can even go as far as designing their own PsyToolkit version (e.g., "myversion-2.0") and keep its functionality entirely separate of the other PsyToolit versions on the same computer. Using this version control technique, researchers do not need to be worried about older experiments no longer being compatible with the newest PsyToolkit version.

Below, four examples demonstrate details of the scripting language. These examples range from simple to complex. The first two examples can be run with the regular keyboard and mouse, whereas the last two examples illustrate the use of external devices. The latter two examples are of potential use beyond PsyToolkit, because they describe how to build simple circuits for using LEDs and measuring light.

EXAMPLE 1
Simple Stimulus–Response Compatibility Experiment

Stimulus–response compatibility studies have demonstrated that people typically respond faster and less erroneously to stimuli that have a straightforward relationship with their associated responses (Prinz & Hommel, 2002). For example, if participants must associate the visually presented words "left" and "right" to a left- and a right-positioned response button, and if the words can appear at any location on the screen, people will respond fastest when both stimulus and response button are located on the same side.

Example 1 (Appendix A) implements a stimulus–response compatibility experiment and aims to illustrate some key concepts of the PsyToolkit scripting language. The instruction for the participant is to respond to red stimuli with the left shift button and green stimuli with the right shift button. Stimuli can appear on the left or right of the screen, but this variation in stimulus position is irrelevant from the participant's point of view. As previously mentioned, the code consists of various sections—in this case, "options" (lines 9–10), "table" (lines 12–17), "task" (lines 19–35), and "block" (lines 37–40), and these sections are separated by empty lines (lines 11, 18, and 36).

The "options" section has just one option (line 10), which specifies that position 0,0 refers to the center of the screen (by default, 0,0 refers to the top left corner of the screen). The "table" section has four rows, each row describing one of the four experimental conditions. The columns reflect the name of the condition, the horizontal position of the imperative stimulus, the red and green color channels of the stimulus, and the correct response, respectively.

The "task" section describes what happens in one trial. The "table" (line 20) command tells the computer to use the table "compatibilitytable" and selects one of its rows for the present trial. In this example, PsyToolkit uses the default method of choosing a table row randomly (each trial), but other methods are available as well. The "keys" command (line 21) states that the left and right shift keys of the keyboard will be used, and these will be numbered as they occur in this line (i.e., 1, 2). A small white rectangle is displayed at the center of the screen (line 22) for 200 msec (line 23) and is then erased (line 24). The rectangle command code takes a number of arguments; the first two are the position, followed by the width and height, followed by the values of the RGB (i.e., red–green–blue) channels (255,255,255 equals white). Following a 50-msec pause (line 25), the "show" command (line 26) is then used for displaying the imperative stimulus, a red or green rectangle. Now, some of the arguments are read from the table, as indicated by the "@" sign (the horizontal position and the red and green values vary from trial to trial).

The "readkey" command (Line 27) waits for up to 5,000 msec for keyboard input, but it will ignore any key except the keys specified in line 21. The second argument of "readkey" specifies what constitutes a correct response, and in this case this is read from the fifth column of the table (as specified by the "@5"), using the randomly selected row of the present trial. Once the "readkey" command finishes, a number of variables related to the "readkey" event will automatically be set. The KEY variable will be set to the number of the keypressed (1 or 2), the STATUS variable is set to CORRECT, WRONG, or TIMEOUT, and finally, the RT and TT variables will be set to the time (since the start of "readkey") the button is pressed down and the time the button is released, respectively. The "clear" command (line 28) erases the second presented stimulus (i.e., the one on line 26) once the response is measured or once the 5,000 msec have passed (a "clear" command can also be called with negative numbers to indicate the relative number of stimulus presentation: −1 would indicate that the last presented stimulus must be erased; see also Example 3).

The "if" statement (line 29) illustrates the PsyToolkit conditional statement; the lines between "if" and "fi" are carried out only if the "STATUS != CORRECT" condition[4] applies.

Finally, the "save" statement is used to save variables permanently to a computer file for later data analysis (the

file name is a combination of the script name and the date of the experiment). Variables are saved in plain text format and can thus easily be read in by any major statistics program. The programmer determines exactly which data are saved and which not. This prevents intermediate data-filtering stages in which the researcher needs to filter out the data of interest (as is the case in some other software for running experiments).

EXAMPLE 2
Visual Search Using the Mouse

Visual search experiments serve the study of a number of cognitive processes, ranging from studies of attention to studies of object representation (Wolfe, 2003). Typically, the participant has to indicate whether or not a target stimulus is present among a set of nontargets. The necessity of the inclusion of nontarget trials can be overcome by using a pointing device: The participant needs to "click" the found target, instead of merely indicating target presence. This can reduce the number of trials significantly and thus improve the efficiency of the experimental visual search task design. But procedures for handling and reading a pointing device are typically more complicated in programming languages than are those for handling a keyboard. Therefore, PsyToolkit includes a number of functions to simplify the use of the mouse, as illustrated in a visual search experiment (Appendix B).

In this example experiment, four stimuli are displayed, and the instruction is to find the red circle.[5] The participant has to move the mouse cursor to the target stimulus and click the left mouse button.

In order to use the mouse, the "options" section must contain the "mouse on" statement (Appendix B, line 3). The "bitmaps" section is used to load bitmaps (lines 6–10) and specifies the names of the bitmap files. PsyToolkit supports all common bitmap formats (e.g., jpeg, tiff, png, bmp) and treats the png file format as default: On line 10, one does not need to specify the complete filename (with extension) to indicate that there is a "redsquare.png" file. In analogy to bitmaps, PsyToolkit can load sound files into memory using the "sounds" section (lines 12–13).

This example experiment uses only four visual stimuli (lines 6–9) and two possible experimental conditions, specified in the table (lines 15–17). In the task section (lines 19–35), first the "searchtable" table is associated with the "searching" task, such that the columns referred to by "@" correspond to the columns of that table (for simplification, the table has only two table rows, specifying two possible stimulus configurations). In line 21, a fixation point is put at the center (position 0,0 is the default position and does not need to be specified explicitly). In order to present multiple stimuli at exactly the same time, rather than one for one, the "show bitmap" statements (lines 24–27) are within a "draw off"–"draw on" part of the code, which guarantees that the computer does not display until it reaches the "draw on" statement.

The "readmouse" (line 29) command takes three or five arguments. The "l" specifies that the left mouse button must be pressed, the "2" specifies that the mouse must be in the area of the second presented bitmap (line 24) in the present "task section," and the 5,000 indicates the maximum response time in milliseconds. The optional "range" element of the "readmouse" command indicates that only mouse clicks on bitmaps 2 through 5 (lines 24–27) will be considered. The latter statement helps to ensure that participants are not punished for clicking the mouse on something that is irrelevant (such as the fixation point, which in this case is bitmap 1).

Once the participant has clicked, all bitmaps are erased at once, and if the participant selected the correct bitmap, a sound will be played. At the end of the "task" section, some of the relevant variables will be saved (line 35). RT is the time the mouse button was pressed, and TT when it was released. MOUSE_X and MOUSE_Y refer to the cursor position when the mouse was clicked (not all off these variables are necessary for a real visual search experiment but are shown merely for illustration).

EXAMPLE 3
LED

In Examples 1 and 2, and in most ongoing studies in experimental psychology, a computer screen is used for the display of visual stimuli. The downside of regular computer screens is that they cannot be controlled with millisecond resolution. For example, on a typical computer monitor of 60 Hz, the display can be changed only every 16.7 msec.

For studies that require stimulus control at a millisecond resolution, the use of LEDs can be a solution. The skills required to set up a small LED display are minimal, and one can even circumvent the need to solder by using a solderless breadboard. LEDs are inexpensive and can be bought at electronic hardware stores (such as RadioShack in the U.S. and Maplin in the U.K.). The more expensive hardware needed is a digital input/output board, such as the parallel port or Measurement Computing's PCI-DIO24. The required items for Example 3 are the PCI-DIO24 card and peripherals, 1 LED (any color, 5-V), a 470-Ω resistor, two flexible jumper wires (ideally, 15–20 cm), and one breadboard.

The PCI-DIO24 device should be installed as instructed by the manufacturer, with a cable (Measurement Computing C37FFS-5) and a 37 universal screw terminal pinboard (Measurement Computing CIO-MINI37). The PCI-DIO24 device requires a driver (a piece of software interfacing the operating system and PsyToolkit), which is distributed for free and is easy to install.[6]

The advantage to the use of a breadboard and a screw terminal is that the whole setup can be created with only a screwdriver as a tool (Figure 1). The LED and the resistor are pushed into the breadboard, and so are the two jumper wires. The two wires are attached to the screw terminal pins 37 (port A0) and 21 (ground).

Once the hardware has been prepared, the script (Appendix C) should be used to run the study. The script is designed to measure the time it takes a participant to respond (i.e., pressing the space bar) to the LED light. There are different intervals between trials, as coded in

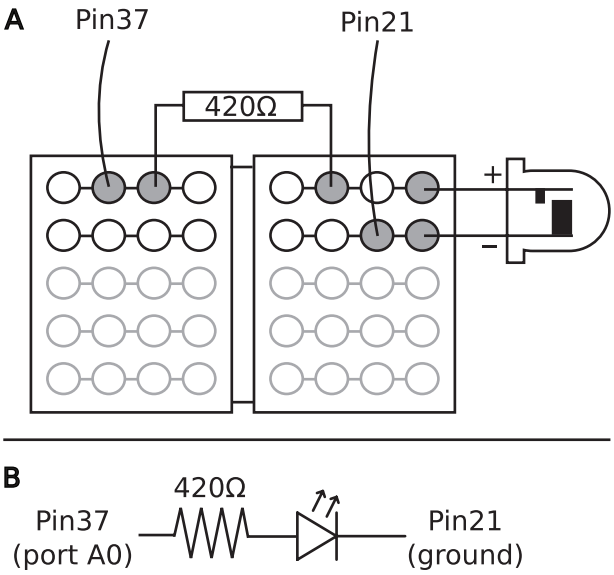

Figure 1. Light-emitting diode (LED) setup. (A) Diagram of the electronic components. Jumper wires and components are pushed into the holes of the breadboard (gray filled circles are here the ones used). The right component is the LED, with an illustration of its positive (cathode) and negative (anode) sides (which can easily be recognized when looking at the transparent bulb of the LED). The component at the top is a resistor. This diagram should be connected to the pins of the terminal pinboard CIO-MINI37, which itself is connected to the PCI-DIO24 digital input/output board. (B) Exactly the same as panel A, but now represented as a conventional electronic diagram.

the table (lines 4–7), which is associated with the "simple-responsetask" task (line 10). To switch an output line of PCI-DIO24 to high, the "pcidio24" command is used with three arguments. The eight lines of port A are coded as a number between 0 and 255 (bit 1 corresponds to line 1 of port A, bit 2 to line 2, etc.). The "readkey" statement (line 13) follows the same logic as in Example 1. To switch the LED off, the digital channel is set to zero (line 14). All other code is similar to that in Examples 1 and 2.

The script is simple and requires elaboration for use in a real experiment, but it illustrates the ease of controlling an LED setup using PCI-DIO24. Up to 24 LEDs can be attached.

EXAMPLE 4
Light Sensor

Light sensors are relatively simple to build and can be very useful for measuring behavior or for verifying the timing accuracy of screens and projectors. Yet there is a lack of simple documentation on how to build a light sensor for this purpose; this example explains how to build one (Figure 2) and how to use it with PsyToolkit and PCI-DIO24 (Appendix D).

The necessary components for the light sensor are a photodiode and a mega-ohm variable resistor (Figure 2). Although the light sensor can be constructed using a breadboard, it is highly recommended to make it more stable by soldering the components together.

The script (Appendix D) introduces some new PsyToolkit features. First of all, there is a "while" loop employed in the "task" section. The script uses "timers" to keep track of the lapsed time. The script uses various variables to keep track of the status of the light sensor. A "C" expression is used, and text is displayed. Below, each of these features will be explained.

Typically, the task section describes just one trial, and it is the "block" section that organizes the looping through a sequence of trials. But in experiments in which the researcher wants to execute many measures that are not necessarily separate trials, it might be more practical to use a "while" loop to carry out multiple measures within this loop. For example, this is useful for implementing a synchronous tapping task (Repp, 2005), in which a participant must repeatedly press a button in response to a regularly flashing LED. The "while" loop can also be useful for certain dual-task setups in which multiple-response events need to be controlled independently. In such a situation, the user can combine the "while" loop with the more advanced "status" arguments of keyboards (e.g., "keystatus," "mousestatus," "cedrus status," and "iolab status").

In the present example, a "while" loop lasting 10 sec (lines 12–30) is used to measure whether the light sensor detects changes in light (line 12). In order to prevent the computer from running the "while" loop faster than necessary, the "nap" command must be included (Finney, 2001).

The "set" command is used to set the value of variables. PsyToolkit distinguishes between local variables (prefixed by a "$" sign) and global variables (prefixed by an "&" sign). Local variables are valid only within the task section they are being used in, whereas global variables can be used and kept between tasks and blocks.

"Timestamp" variables are used to store the current time in a special class of time variables (lines 10, 11, and 13). The difference between two "timestamps" can be calculated and stored in a variable using the set command in combination with the "timestamp-diff" argument (line 14).

Occasionally, "C" syntax can be useful for more complex expressions, such as the "logical and" expression.

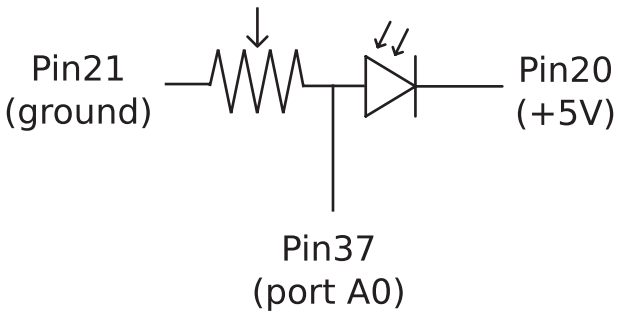

Figure 2. Diagram of a light sensor. The two components of this diagram are a variable resistor (1 MΩ) and a photodiode. This circuit should be connected to the pins of the terminal pinboard CIO-MINI37. The variable resistor can be adjusted to light conditions.

The A0 port of the PCI-DIO24 device contains the value 1 or 0, depending on whether light is shining on the sensor or not. Because only the A0 port is used (i.e., one line of the eight lines of port A), the other ports (A1–A7) are irrelevant, and the value of the A0 port can be filtered out from the other values port A can hold using the logical "and" command (line 16).

When this example script is running, the screen should display "light is on" when the light goes on and "light is off" when the light goes off. Text display is performed using the "show text" command, which per default uses the first font loaded ("Times New Roman, 40 pt," line 6).

DISCUSSION

In summary, PsyToolkit is a well-developed and -tested software package for programming psychological experiments on Linux computers. PsyToolkit is suited not only for technically oriented users, but also for those on a limited budget, because both the operating system Linux and Psytoolkit are free of cost. The four examples, ranging from simple to complex, have reviewed and explained a variety of functions. Altogether, this review gives potential users a good basis on which to start experimenting with PsyToolkit.

The remainder of this review addresses limitations of this software and plans for future work. One limitation of PsyToolkit is that it is available only for Linux. In principle, it is possible to port PsyToolkit to other platforms, especially since it uses software libraries that are available for the common computing platforms. The most difficult aspect of such a port is the programming of the Linux specific interface library for the Measurement Computing board. Furthermore, much of the development time spent on PsyToolkit was devoted to testing time-critical functions and hardware compatibility. A complete port to other computer platforms is possible, but it would take considerable effort to ensure that the complete package is ported (not just the parts that are easy to port) and to submit the software to the same level of testing. Currently, there are no plans to work on porting PsyToolkit, but anybody with a strong desire to use PsyToolkit should consider that Linux is reasonably easy to install (and Linux can easily be added to computers that already have another operating system installed).

Another limitation of PsyToolkit is that it does not come with the same support users can expect from commercial packages—there is no phone hotline or support team.

PsyToolkit will be further developed with a focus on two aims. First of all, the reliability of this software has priority over the development of new features or porting to other platforms. Second, it is intended to release PsyToolkit in more Linux-distribution-specific "packages" to ease the installation (currently, a custom source-based installer and a Debian/Ubuntu package are available).

AUTHOR NOTE

The author thanks IoLab Systems Inc., Cedrus Inc., Warren J. Jasper of North Carolina State University, and Lawrence H. Snyder of Washington University in St. Louis for support. Support for two PCI24 boards was made possible by a grant from the Experimental Psychology Society. Correspondence concerning this article should be addressed to G. Stoet, Institute of Psychological Sciences, University of Leeds, Leeds LS2 9JT, England (e-mail: g.stoet@leeds.ac.uk).

REFERENCES

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

GELLER, A. S., SCHLEIFER, I. K., SEDERBERG, P. B., JACOBS, J., & KAHANA, N. J. (2007). PyEPL: A cross-platform experiment-programming library. *Behavior Research Methods*, **39**, 950-958.

GOFEN, A., & MACKEBEN, M. (1997). An introduction to accurate display timing for PCs under "Windows." *Spatial Vision*, **10**, 361-368.

MCKINNEY, C. J., MACCORMAC, E. R., & WELSH-BOHMER, K. A. (1999). Hardware and software for tachistoscopy: How to make accurate measurements on any PC utilizing the Microsoft Windows operating system. *Behavior Research Methods, Instruments, & Computers*, **31**, 129-136.

PEIRCE, J. W. (2007). PsychoPy—Psychophysics software in Python. *Journal of Neuroscience Methods*, **162**, 8-13.

PLANT, R. R., HAMMOND, N., & TURNER, G. (2004). Self-validating presentation and response timing in cognitive paradigms: How and why? *Behavior Research Methods, Instruments, & Computers*, **36**, 291-303.

PLANT, R. R., HAMMOND, N., & WHITEHOUSE, T. (2002). Toward an Experimental Timing Standards Lab: Benchmarking precision in the real world. *Behavior Research Methods, Instruments, & Computers*, **34**, 218-226.

PLANT, R. R., HAMMOND, N., & WHITEHOUSE, T. (2003). How choice of mouse may effect response timing in psychological studies. *Behavior Research Methods, Instruments, & Computers*, **35**, 276-284.

PRINZ, W., & HOMMEL, B. (2002). *Common mechanisms in perception and action: Attention and performance XIX*. Oxford: Oxford University Press.

R DEVELOPMENT CORE TEAM (2009). *R: A language and environment for statistical computing* [Computer software]. Vienna: R Foundation for Statistical Computing. Available from www.R-project.org.

REPP, B. (2005). Sensorimotor synchronization: A review of the tapping literature. *Psychonomic Bulletin & Review*, **12**, 969-992.

STAHL, C. (2006). Software for generating psychological experiments. *Experimental Psychology*, **53**, 218-232.

STEVENS, M., LAMMERTYN, J., VERBRUGGEN, F., & VANDIERENDONCK, A. (2006). Tscope: A C library for programming cognitive experiments on the MS Windows platform. *Behavior Research Methods*, **38**, 280-286.

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

STRAW, A. D. (2008). Vision Egg: An open-source library for realtime visual stimulus generation. *Frontiers in Neuroinformatics*, **2**(4).

WOLFE, J. M. (2003). Moving towards solutions to some enduring controversies in visual search. *Trends in Cognitive Sciences*, **7**, 70-76.

NOTES

1. http://psytoolkit.leeds.ac.uk.

2. The C library itself uses other common libraries, especially the SDL libraries, but also the hid and parapin libraries.

3. In fact, IoLab is a complex device integrating keyboard, voicekey, and digital input/output.

4. "!=" stands for "not equal."

5. This example uses only four stimuli, but increasing the numbers of stimuli and trials would be trivial.

6. This driver is written by W. J. Jasper, North Carolina State University, and can be downloaded from ftp://lx10.tx.ncsu.edu/pub/Linux/drivers/.

APPENDIX A
PsyToolkit Scripting Code for Example 1

```
 1 # text following hash marks is comment for humans only (if you do not
 2 # count lines that are entirely comments, there are 31 lines of real
 3 # code)
 4 # simple stimulus response compatibility study with 40 trials
 5 # instruction for participants:
 6 #  press left shift button in response to red stimuli
 7 #  press right shift button in response to green stimuli
 8
 9 options
10 centerzero # position 0,0 is center of screen
11
12 table compatibilitytable # each row describes 1 of the 4 conditions
13 #-name of condition----------------position--red?--green?--correct key------
14  "leftresponse compatible"           -200     255   0       1
15  "leftresponse incompatible"          200     255   0       1
16  "rightresponse compatible"           200       0   255     2
17  "rightresponse incompatible"        -200       0   255     2
18
19 task compatibility
20  table compatibilitytable            # table to be used
21  keys lshift rshift                  # the two keys (numbers 1 and 2) to be used
22  show rectangle 0 0 10 10 255 255 255 # small white rectangle at
23  delay 200                           # center for 200 ms
24  clear 1                             # erase the rectangle
25  delay 50                            # and wait 50 ms
26  show rectangle @2 0 100 100 @3 @4 0  # target stimulus is presented
27  readkey @5 5000                     # wait for key of column 5 of table
28  clear 2                             # remove target stimulus from screen
29  if STATUS != CORRECT                # check the result of the "readkey" command
30    show rectangle 0 0 300 300 255 0 0 # show big central red rectangle
31    delay 1000
32    clear 3                           # remove the third stimulus shown
33  fi                                  # "fi" ends the "if" part
34  delay 1000                          # wait 100ms (inter trial interval)
35  save BLOCKNAME @1 KEY STATUS RT     # save data to file
36
37 block test                           # this block is called "test"
38  tasklist                            # list the tasks being used, here we use only 1
39    compatibility 40                  # run the task 40 times
40   end                                # end of the "tasklist" part
```

APPENDIX B
PsyToolkit Scripting Code for Example 2

```
options
  centerzero
  mouse on

bitmaps
  fixpoint fixpoint.jpg
  redcircle redcircle.bmp
  greensquare greensquare.tiff
  greencircle
  redsquare

sounds
  soundsok sayingok.wav

table searchtable
  redcircle -100 -100 greensquare 100 100 redsquare -100 100 greencircle 100 -100
  redcircle 100 100 greensquare -100 100 redsquare 100 -100 greencircle -100 -100

task searching
  table searchtable
  show bitmap fixpoint
  delay 100
  draw off
    show bitmap @1  @2  @3
    show bitmap @4  @5  @6
    show bitmap @7  @8  @9
    show bitmap @10 @11 @12
  draw on
  readmouse 1 2 5000 range 2 5
  clear 1 2 3 4 5
  if STATUS == CORRECT
    sound soundsok
  fi
  delay 2000
  save TABLEROW STATUS RT TT MOUSE_X MOUSE_Y

block searchblock
  tasklist
    searching 5
  end
```

APPENDIX C
PsyToolkit Scripting Code for Example 3

```
options
 pcidio24 out a                    # setup the PCIDIO24 board

table intertrialintervaltable      # a table with 3 rows/conditions
 "short"   500
 "medium"  1000
 "long"    2000

task simpleresponsetask
 table intertrialintervaltable     # table to choose conditions from
 keys space                        # only one response key being used, the space bar
 pcidio24 set a 1                  # LED on (controlled by bit 1 of port A)
 readkey 1 3000                    # wait for pressing the space bar for 3 seconds
 pcidio24 set a 0                  # LED off
 if STATUS == CORRECT              # check the result of the "readkey" command
  show rectangle 0 0 300 300 0 255 0  # show big central green rectangle
  delay 250                        # wait 250 ms
```

APPENDIX C (Continue)

```
18    clear -1                          # remove the last stimulus shown
19   fi                                 # "fi" ends the "if" part
20   delay @2                           # delay (intertrial time)
21   save BLOCKNAME @1 @2 RT            # save data to file
22
23 block testdio24                      # this block is called "testdio24"
24   tasklist                           # list the tasks being used, here we use only 1
25     simpleresponsetask 10            # run 10 trials

26    end                               # end of the "tasklist"
```

APPENDIX D
PsyToolkit Scripting Code for Example 4

```
 1 options
 2   pcidio24 in a
 3   centerzero
 4
 5 fonts
 6   times times.ttf 40 # make sure the file "times.ttf" exists!
 7
 8 task lightsensor
 9   show text "The light is off"
10   timestamp TimeBegin
11   timestamp TimeNow
12   while $totaltime < 10000
13     timestamp TimeNow
14     set $totaltime timestamp-diff TimeBegin TimeNow
15     set $lightstatus pcidio24 a
16     set $lightstatus c-expression lightstatus & 1
17     if $lightstatus != $previousstatus
18       if $lightstatus == 1
19         clear -1
20         show text "light is on "
21       fi
22       if $lightstatus == 0
23         clear -1
24         show text "light is off "
25       fi
26       save $lightstatus $totaltime
27       set $previousstatus $lightstatus
28     fi
29     nap
30   while-end
31
32 block measurelight
33   tasklist
34     lightsensor 1
35   end
```

(Manuscript received March 9, 2010;
revision accepted for publication June 15, 2010.)