Research article

Software wizards to adjust keyboard and mouse settings for people with physical impairments

Heidi Koester[1], Richard Simpson[2], Jennifer Mankowski[1]

[1]Koester Performance Research, Ann Arbor, MI, USA, [2]University of Pittsburgh, Pittsburgh, PA, USA

Context/objective: This study describes research behind two software wizards that help users with physical impairments adjust their keyboard and mouse settings to meet their specific needs. The Keyboard Wizard and Pointing Wizard programs help ensure that keyboard and pointing devices are properly configured for an individual, and reconfigured as the user's needs change. We summarize four effectiveness studies and six usability studies.

Methods: Studies involved participants whose physical impairments affect their ability to use a keyboard and mouse. Effectiveness studies used an A-B-A design, with condition A using default Windows settings and condition B using wizard-recommended settings. Primary data were performance metrics for text entry and target acquisition. Usability studies asked participants to run through each wizard, with no outside guidance. Primary data were completion time, errors made, and user feedback.

Results: The wizards were effective at recommending new settings for users who needed them and not recommending them for users who did not. Sensitivity for StickyKeys, pointer speed, and object size algorithms was 100%. Specificity for StickyKeys and pointer speed was over 80%, and 50% for object size. For those who need settings changes, the recommendations improved performance, with speed increases ranging from 9 to 59%. Accuracy improved significantly with the wizard recommendations, eliminating up to 100% of errors. Users ran through the current wizard software in less than 6 minutes. Ease-of-use rating averaged over 4.5 on a scale of 1 to 5.

Conclusion: The wizards are a simple yet effective way of adjusting Windows to accommodate physical impairments.

Keywords: Disability, Assistive technology, Computer use, Spinal cord injuries, Multiple sclerosis, Amyotrophic lateral sclerosis, Employment, Keyboard, Typing

Introduction

Importance of computer access

There are approximately 50 million people in the US with a disability, 21.5 million of whom are working-age adults (age 21–64 years).[1] About 1 million of these are individuals who have traumatic spinal cord injury (SCI), multiple sclerosis (MS), or amyotrophic lateral sclerosis (ALS).[2–4] Statistically, having a disability has major negative consequences in terms of education, employment, and economic status. In 2005, 25% of working-age people with disabilities had not completed high school vs. 12% of people without disabilities.[1] Only 13% of working-age people with disabilities had at least a bachelor's degree, as compared with 30% of those without disabilities.[1] In 2009, working-age people with disabilities were only 22% as likely to be employed as persons without disability,[5] and this gap in employment has actually gotten worse over the last 20 years. Finally, in 2008, working-age people with disabilities were three times more likely to be living at or below the poverty line.[5]

Although many factors contribute to this situation, effective computer use does have a significant role in reducing these gaps. Although it is difficult to precisely count the number of people whose physical impairments interfere with their ability to use a "standard" keyboard or mouse, about 3% of the US population (about 9 million people) report difficulty typing,[6] and 6.8 million working-age computer users report a severe

Correspondence to: Heidi Horstmann Koester, Koester Performance Research, 2408 Antietam, Ann Arbor, MI 48105, USA.
Email: hhk@kpronline.com

© The Academy of Spinal Cord Injury Professionals, Inc. 2012
DOI 10.1179/2045772312Y.0000000049

dexterity difficulty.[7] For those with manual dexterity impairments, effective access to a computer is a key to accessing educational curricula and performing school work in a productive manner. Computer use helps compensate for many types of disabilities and increases the possibilities for productive employment.[8] Most traumatic SCI occurs in young adults with many productive workforce years ahead of them.[2] Individuals with spinal cord disease such as MS and ALS may still be well within working age by the time their disease causes significant impairments. Computer use has been shown to speed return-to-work and enhance earnings for people with SCI.[8] Looking to the future, most well-paying new jobs will require both a high level of education and significant computer skills.[9]

Easy, productive access to computers is also critical for enhancing the quality of life and independent pursuit of goals for people with disabilities.[10,11] Computer use has been shown to contribute to improved health status by providing access to health information and interaction with clinicians and peers.[6] It can also reduce social isolation by eliminating physical barriers, facilitating communication, and providing a community forum.[12]

The need for improved user-device matching in computer access

Without appropriate accommodations, computer use may be impossible or much slower and uncomfortable than it needs to be. In contrast, appropriate accommodations support productive and comfortable computer access, providing more equal access to educational, vocational, and leisure opportunities. It is critical that the computer system be closely matched to the user's needs and abilities. An important part of this matching process is configuring the user's computer input devices to appropriately leverage user strengths and accommodate limitations.

Current approaches to configuration

A computer input device is typically configured for a user in one of three ways. The first, and most common, is to use the default values for the device. Default values that are not appropriate for a user's abilities may result in difficulty selecting targets with the mouse, decreasing user performance and satisfaction. In a more extreme case, the system may be virtually unusable under the default values.

A second scenario is having the user make his/her own adjustments. This requires that the user knows what parameters are available and how to adjust them. This is a complex task. Performing all possible adjustments for the mouse within Windows XP requires accessing two separate Control Panel applications with six tabbed panels, while making objects larger for easier selection would require accessing a number of additional Control Panel applications. Even if the user can successfully navigate the options, knowing the most appropriate values for all applicable settings may be even more difficult. Users may not understand how the parameter settings relate to the interface problems they are having, or if they do, the best choice of specific values may be unclear. Finally, some options, such as the double-click distance, are not even available for user adjustment. In Windows 7, moving accessibility settings into the Ease of Access Center has not changed the fundamental challenge of making effective changes quickly and easily.

There are some tools available to help users navigate this process. Information modules, such as the 'My Computer My Way' website (http://www.abilitynet.org.uk/myway), try to give clear information about what settings are available and how to use them properly. The Windows operating system includes an accessory program called the Accessibility Wizard, which provides some help in reducing the complexity of configuration for keyboards, pointers, and the visual display. However, neither of these tools includes all available settings, nor do they give specific suggestions about how to appropriately set parameter values based on user performance. They also cannot adapt to user needs over time; users must manually adjust their settings each time they want to make an adjustment.

A third scenario occurs when a clinician or teacher is available to assist with the configuration process, using clinical observations and knowledge of the possible accommodations as a guide. However, most users with physical disabilities do not have a qualified clinician available to them. Trewin and Pain[13] found that only 35% of 30 computer users with physical disabilities had a "computer teacher." Further, not all helpers have the skills to effectively assist.[14] Even when a clinician or other advisor with relevant expertise is available, input device configuration often requires considerable trial and error, with correspondingly high financial and time costs.

Current approaches are not working

Current methods may lead to appropriate input device configurations in some cases, but it takes special knowledge, time, and continued maintenance to do it right.[10] As a result, input devices are often not appropriately configured to meet users' needs, with consequent negative effects on user productivity and comfort. People

with disabilities still have 25–50% lower computer usage than people without disabilities.[6,7] The reasons are directly related to lack of effective accommodation for impairment: physical difficulty with typing has the strongest association with decreased computer and Internet use.[6]

For those who do use computers, access is often a struggle. For example, in a study of mouse users with physical disabilities, 55% of the dragging tasks and 40% of the double-click attempts were unsuccessful.[15] In another study, pointing device use for subjects with high-level SCI was only 30% as fast as for users without impairments.[11] A third study found that double-clicking required multiple attempts more than two-thirds of the time.[16] For keyboard use, typing speeds below 5 words/minute and error rates above 20% are not unusual.[17]

In many cases, users may not be aware that there are alternatives to the standard interface. Thirty-six percent of computer users with mild or severe disabilities were unaware of the free, built-in accessibility settings offered by the Windows and Mac operating systems, and only about 20% of computer users with disabilities report using some form of computer access technology.[18,19]

Another challenge is that a single configuration may not be appropriate for a user at all times. A user's needs may change due to changes in his/her abilities which may happen over the course of a day (e.g. fatigue) or longer (e.g. due to progression of the disability, recovery of function, or other factors). The user's needs may also change based on the user's desired tasks (e.g. some computer activities may require greater precision than others). Even if a clinician is available to recommend an initial configuration, he/she is unlikely to be available every time adjustments to the configuration are desirable. If a user is responsible for his/her own adjustments, he/she may not notice, or know how to respond to, a gradual decline in performance. Furthermore, interface configuration is secondary to the user's primary computer tasks; even if it can be done effectively, it takes time, physical effort, and cognitive focus away from more central tasks.

The need for automatic configuration

What is missing is an effective way to deploy the solutions that are most appropriate to a user's specific needs. An automated software agent on the user's computer could help ensure that input devices are properly configured for the individual, and reconfigured as the user's needs change. Such an agent would need a means to observe the user's performance and recommend appropriate input device configuration settings based on that performance. Several groups have proposed configuration agents for input device configuration.[20–26]

Most development for input device agents has occurred in the area of keyboard settings. Trewin *et al.* have developed a configuration agent for keyboard settings, called Keyboard Optimizer.[13,24,27–29] The agent creates a user model based on free typing and recommends values for several keyboard settings. The agent's recommendations were evaluated with 20 keyboard users who have physical disabilities, and use of the agent-recommended settings significantly reduced key repeat errors.[27] Effects on productivity measures, such as typing speed, were not measured. This keyboard agent is not currently available to end users, however. We are not aware of any agents available for the automated configuration of pointing devices.

The purpose of this project is to develop software tools for the automatic configuration of keyboard and pointing device settings. This software will accommodate the needs of people with physical impairments, leading to improved productivity and comfort during computer use.

Our approach

Our approach focuses on enhancing computer usage by having the computer take some of the responsibility for adapting to the user's strengths and limitations. By measuring the user's behavior during typical computer use, the computer could diagnose difficulties experienced by the user and recommend effective accommodations to mitigate those difficulties. This significantly reduces the complexity, configuration, and maintenance barriers to users' effective computer use.[30] It also supports the dignity of the individual, by creating a built-in approach to access that meets all users where they are and minimizes the need for "special" procedures for users who have impairments. User autonomy is also preserved, as the decision-making rests with the user.

Each agent, or "wizard", walks the user through several steps:

1. Perform a task (such as typing, or clicking on targets)
2. View the wizard's recommendations, based on task performance
3. Try the recommended settings
4. Choose which settings to activate for continued use.

The goal is to provide the user with settings that allow for more comfortable and productive computer use, within an efficient and easy-to-use process.

Table 1 Keyboard configuration parameters. Parameters in bold have been addressed in this project to date

Parameter	Description
Repeat delay	How long a key must be held down before it begins to repeat
Repeat rate	Once the keyboard begins to repeat a character, the rate at which it repeats
SlowKeys	How long a key must be held down before it is accepted
BounceKeys	Tells the operating system to ignore keystrokes that are depressed within x seconds of the previous key release
StickyKeys	When StickyKeys are activated, the typist can enter key combinations (e.g. Shift-A to type a capital A) by pressing the modifier key (e.g. Shift) and other keys (e.g. "A") in series, rather than holding down multiple keys simultaneously
ToggleKeys	Gives an auditory signal when locking keys, such as Caps Lock, are depressed

Configuring keyboards and pointing devices

Configuring keyboards

There are a variety of keyboards to choose from, ranging from "standard" desktop keyboards, to keyboards with large keys, to mini keyboards. Once a given keyboard is selected, tuning it to the user's strengths and limitations may yield significant performance and comfort benefits. Keyboard behavior can be adjusted within the Windows and Macintosh operating systems using the parameters in Table 1. These settings are included in the operating system, and there may be others for third-party keyboards depending on their design and associated software. The potential consequences of inappropriate settings are many.[10] For example, for someone who types with a mouthstick, not having StickyKeys active makes it cumbersome to type capital letters and impossible to use other key combinations such as shifted punctuation or Ctrl-C. For someone who can target a key reasonably well but has difficulty releasing from the key, the default settings for repeat delay will cause numerous additional characters to appear in documents. In Trewin and Pain's[15] study of keyboard users with physical disabilities, key repeat errors were the most common problem observed, comprising an average of 11% of all keystrokes.

Configuring pointing devices

A "pointing device" is anything that controls the computer's mouse cursor. This includes the standard mouse, as well as alternatives such as trackballs, laptop trackpads, head-controlled pointers, and many other devices. Typical parameters for pointing device configuration within the Windows and Macintosh operating systems are shown in Table 2. Proper adjustment of these settings can be critical to efficient use of a pointing device for people with disabilities. For someone with impaired motor control, the default pointer speed may cause the cursor to move too quickly, making it difficult or impossible to select small targets such as toolbar buttons. Other difficult tasks include double-clicking the mouse button while keeping the mouse still.

Description of the wizard software

We have developed two wizard applications. The Keyboard Wizard provides adjustments to Windows keyboard settings. The wizard walks users through a one-sentence typing task, suggests possible changes to the keyboard settings, and lets the user decide which settings to activate. Similarly, the Pointing Wizard provides adjustments to Windows mouse settings. Both software applications are available at the KPR website (www.kpronline.com).

The Keyboard Wizard currently supports three of the five major Windows settings that control keyboard behavior: repeat delay, repeat rate, and StickyKeys. The algorithms used to make the repeat recommendations are based on Trewin's published algorithms,[24,27] and

Table 2 Built-in configuration options related to pointing devices

Parameter	Description
Button-handedness	Controls the functions assigned to the left and right mouse buttons
Click method	Whether the user performs a single or double click to open icons
Double-click time	Controls the allowable time between two clicks in a double-click
Double-click distance	Controls the allowable distance between two clicks in a double-click
Pointer speed (gain)	How quickly the cursor moves across the screen in response to mouse movements
Enhance pointer precision	The Enhance Pointer Precision setting enables a complex algorithm controlling the velocity and acceleration of the mouse cursor
ClickLock	An alternative drag method. To start a drag, the user presses the mouse button for a set time. A second mouse click releases the drag
Object size	It is possible to increase the size of icons, menu bars, and other objects in the user interface, which may make them easier to select with the mouse.

Parameters in bold have been addressed in this project to date.

rely on measuring how long the user holds down each keypress while typing. The StickyKeys algorithm employs some basic heuristic rules to determine whether and in what manner the user is entering modified characters such as capital letters and certain punctuation. For example, usage of the Caps Lock key, combined with lack of using the Shift key, suggests that the user cannot use the Shift key to create capital letters. This algorithm has also evolved so that it can now detect when pressing two keys simultaneously is possible yet difficult. This is based on our observation that people having difficulty using the Shift key tend to hold it down at least four times longer than they hold other keys down. The algorithm can also recommend turning StickyKeys OFF if someone has it ON unnecessarily. These are by far the most commonly used settings, but we do have future plans to recommend settings for SlowKeys and BounceKeys.

The Pointing Wizard supports six Windows settings that control pointing device behavior. The six supported settings are

1. Pointer speed – also known as gain, this controls how far the mouse cursor moves in response to the user's physical action. With a higher pointer speed, the mouse is more responsive, but can feel too sensitive. With a lower pointer speed, it may be easier to click on smaller targets, but the mouse can feel sluggish. The right gain for an individual depends on their psychomotor control over their pointing device. Our algorithm includes a screening model that assesses whether gain should increase, decrease, or stay the same. If the screen determines that the initial gain is appropriate, no further user action is required. Otherwise, the algorithm begins a dynamic phase which hones in on the user's appropriate gain within a few minutes.
2. Enhance pointer precision (EPP) – this is a Windows setting that supports a two-stage gain control: a lower gain for slower user movements and a higher gain for faster user movements. This gives all users enhanced control over the mouse, and we found in phase I that having this setting ON enhanced pointing performance by an average of 17%.[31] Therefore, our algorithm recommends that EPP always be turned ON. This is a simple rule, but in implementing it, we had to consider the possible interaction between the EPP and pointer speed settings. We analyzed our database of pointing data from users with physical disabilities, and determined that EPP and pointer speed could be recommended independently from each other.
3. Double-click time – this defines how quickly two clicks have to follow each other in order to qualify as a double-click. This is based on measurements of the interval between clicks while the user performs a double-click task.
4. Double-click distance – this defines how far apart two clicks can be in order to qualify as a double-click. This is based on measurements of the distance between clicks while the user performs a double-click task.
5. Use double-click – this defines whether double- or single-clicks are required to open items in Windows Explorer, such as folders, files, and application programs. By default, double-click is used. The algorithm recommends this setting based on the difficulty observed when a user double-clicks on targets.
6. Object size – for some users with physical impairments, larger on-screen items may be easier to select with a pointing device. Larger items tend to make selection faster for everyone, but the effect is especially pronounced for people who have significant difficulty with small targets. The wizard presents both 16 and 32 pixel targets, and the algorithm recommends the use of larger targets when the measured performance benefit exceeds specific thresholds.

Note that within each wizard, we implemented methods that programmatically activate the recommended setting, thus saving the user from having to make changes manually via the Control Panel or other means. This was non-trivial as it involves interaction with the Windows registry.

General hypotheses

The wizards' recommendations must be effective. That is, the algorithms that "decide" what settings best match a user's specific needs must make decisions that benefit users who need changes, and leave those who do not need changes alone.

The wizard interface must be usable by end users. As noted above, many individuals do not have access to assistive technology practitioners who can help them through this process. And in any case, they should not have to rely on a "gatekeeper" for something as fundamental as adjusting Windows settings.

These considerations lead to the following general hypotheses:

1. Effectiveness: Suggestions from the wizard software will significantly enhance user performance on keyboard and mouse tasks.
2. Usability: End users with physical impairments will be able to navigate the wizards' user interface independently, easily, and efficiently.

These hypotheses were tested in a variety of effectiveness and usability studies, as described below.

Effectiveness studies

Summary of previous work

Keyboard wizard algorithms

In a study of 12 typists with physical impairments, our algorithms recommended repeat rate, repeat delay, and

use of StickyKeys, after users performed a prescribed typing task.[17] Use of the recommended settings significantly improved text entry rate and typing accuracy. Two participants had significant problems with inadvertent key repeats, when using the default repeat settings. For those two participants, use of the wizard-recommended repeat settings reduced the number of repeated characters by 96% and significantly improved text entry rate by 36%. The Keyboard Wizard recommended StickyKeys for six participants, each of whom had at least one problem related to modifying keys without StickyKeys. Use of StickyKeys for these individuals eliminated their modifier-related errors and significantly improved typing speed by 14%. The Keyboard Wizard did not recommend StickyKeys for the six participants who demonstrated no need for it.

The StickyKeys algorithm has evolved to better handle cases where modifying keys is difficult but not impossible for the user. The algorithm considers not just whether a modifier key was successfully entered, but how long it took to do so. We examined this algorithm's performance on data from 35 individuals: 10 people with and 25 without physical impairments. The specificity was 22/25, or 88%, so the algorithm was effective at not recommending StickyKeys for those who do not need it. The sensitivity was 10/10, or 100%, which means the algorithm correctly identified all those who are appropriate StickyKeys users.

Pointing wizard algorithms

To examine our algorithms that recommend appropriate pointer speed (gain) settings for users of all pointing devices, we conducted two studies involving 22 participants with disabilities.[32,33] Approximately 25% of participants received a meaningful performance enhancement when using the algorithm-selected gain as compared with the operating system's default gain. For the other participants, the algorithm-selected gain neither helped nor hurt performance. While this suggests that gain is not a strong factor in pointing performance for most users, it does point out the potential value in identifying those who may benefit from a change and adjusting their gain accordingly, especially if recommendations can be made without inconveniencing those users who do not need an adjustment.

We have also built and tested methods for recommending double-click time and double-click distance.[16] Twelve individuals with physical impairments used these algorithms to get recommendations for double-click settings that matched their abilities. The algorithm recommended new double-click settings for eight participants; the remaining four required no change. Subsequent use of the recommended settings significantly enhanced subjects' speed and accuracy when double-clicking on targets with their pointing device, leading to an average of 33% fewer clicks per target, 17% faster target acquisition time, and a 29pp improvement in error-free targets (all significant at $P < 0.01$). The effect was very pronounced in some cases, such as one subject who selected 97% of targets with no errors using the recommended settings, but only 31% without errors using the default settings.

This same study also involved targets of 16 pixels and 32 pixels, as a means of investigating how much larger objects would enhance acquisition time. For object size, 32 pixel targets averaged 19% faster acquisition time as compared to 16 pixel targets (significant at $P < 0.05$). The effect ranged from a minimum of 1% to a maximum of 30%. This effect was also observed in the second part of the study, in which subjects performed real-world Windows tasks with smaller vs. larger targets. In those tasks, the maximum benefit from larger targets was a 38% enhancement in acquisition time.

Follow-up study on pointing device settings

Hypotheses

This study further examined recommendations for pointer speed, as well as double-click settings and object size. The hypotheses were

1. The Pointing Wizard can give appropriate guidance about a user's pointer speed setting, thereby improving a user's ability to successfully acquire targets with their pointing device;
2. The Pointing Wizard can give appropriate guidance about a user's double-click settings, thereby improving a user's ability to double-click successfully with their pointing device;
3. The Pointing Wizard can give appropriate guidance about whether to increase target size, thereby improving a user's ability to successfully acquire targets with their pointing device.

Methods

Twelve individuals with physical impairments participated. The protocol involved two parts. In part 1, users completed three target acquisition tests in an A-B-A design. In the first A condition, subjects were asked to single-click on 16 targets, then double-click on 16 targets, presented one at a time. The gain and double-click settings were set to the Windows default values. Targets were 16-pixel and 32-pixel squares, presented at random locations on the screen. From data in the first A condition, the software derived

recommendations for the user's double-click settings. Subjects selected an additional 32 targets at various gain settings, in order to generate a gain recommendation for each subject. In the B condition, subjects selected 32 targets using the gain and double-click settings chosen by the Pointing Wizard. Finally, in the reversal A condition, they selected another 32 trials using the default Windows settings.

In part 2, subjects performed four sets of seven Windows tasks, including minimize and maximize a window, use the scroll bar, and select from a menu. The purpose was to examine gain and object size effects in more "real-world" tasks. The first and last sets served as baseline A conditions, using default gain and object sizes. The second set used the recommended gain with default object size, and the third set used the recommended gain with larger objects.

In part 1, target acquisition time, number of entries per target, and % of error-free targets were measured by the software. One entry was counted each time the mouse cursor entered the target, so every successful target selection has a minimum of one target entry. An error-free target was one in which only one click was required to acquire the target. In part 2, acquisition time and number of entries per target were counted manually by analyzing video recordings of the tasks. Data were analyzed for A-B-A reversal for each subject. Group statistics were performed using paired *t*-tests between the pooled A (default) conditions and the B (Pointing Wizard) condition.

Results

Gain settings

The effect of gain was fairly neutral, neither very helpful nor harmful. Across all participants in part 1, the recommended gain did modestly enhance target acquisition time, by an average of 9.2% (significant at $P < 0.05$). Two individuals showed meaningful reversals for entries, in which there were 41% fewer entries with the recommended gain as compared with both default conditions. In part 2, with more real-world tasks, there was no significant effect of gain on target acquisition time or entries, for the subject group as a whole. Three individuals showed meaningful reversals for entries in part 2, in which there were 59% fewer entries with the recommended gain as compared with both default conditions. This included the same two people with meaningful reversals in part 1.

Fig. 1 shows the effect of gain on target entries for one subject in part 1 of the study. This person had cerebral palsy, and uses a trackball with her foot. Control at the default Windows setting was fairly difficult for her,

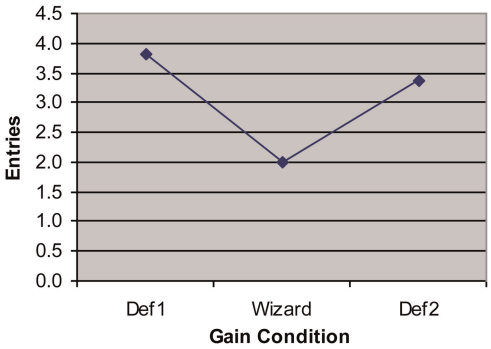

Figure 1 Target entries for one participant when using Windows default gain (Def1, Def2) and the wizard-recommended gain (Wizard).

as reflected in the average target entries of 3.5. The Pointing Wizard recommended a gain of 2 for her, which decreased the target entries to 2.0. The lower sensitivity makes target acquisition a lot easier for her (although the time improvement was only about 9% better with the Pointing Wizard).

Screening score for gain recommendations

We were also able to evaluate the screening score that the algorithm uses to help determine a gain recommendation for a user. After the first eight target acquisition trials, the algorithm computes a screening score to determine whether to try a higher or lower gain in the next set of eight trials.[32] In originally deriving the equation for this screening score, we noticed that almost everyone who needed a higher gain had a screening score below 0.4, and almost everyone who needed a lower gain had a screening score above 0.6. In fact, for 74 cases where a gain change was needed, only 6 had a screening score between 0.4 and 0.6. This suggests that a score in this range could be used to screen out individuals who do not need any gain adjustments, thus saving them the extra time of needlessly selecting additional targets during use of the Pointing Wizard.

For the 12 participants in this study, we classified each regarding whether a change in gain seemed appropriate or not. This was based on each person's performance at four different gain settings, which was measured during the Pointing Wizard's determination of the recommended gain for each person. Six people were classified as not requiring a change in gain ("No"), because their performance either did not change notably for different gains, or was best with their starting gain setting. The other six were classified as potentially benefitting from a change in gain ("Yes"). We then used the screening score to do a similar classification, where scores between 0.35 and 0.6 meant that no change was needed ("No"), and scores outside that range meant

Table 3 Classification of whether subjects' pointing performance might benefit from a change in gain

		Classification based on performance	
		Yes	No
Classification based on screening score	Yes	6	1
	No	0	5

that a gain change might be beneficial ("Yes"). The classifications from the screening rule matched those based on actual performance for 11 of 12 participants. The specific results are shown in Table 3. This translates to a specificity of 5/6, or 83%, and a sensitivity of 6/6, or 100%. This provides confidence in the discriminatory ability of the screening rule, so we implemented it into the Pointing Wizard. Note that passing the screening score does not guarantee that gain will provide a benefit; it simply gives the algorithm an indication that adjusting gain is worth exploring.

Double-click settings

For double-click settings, results were remarkably consistent with those from the earlier pointing study. Eleven of 12 subjects received non-default double-click recommendations. Use of the double-click settings recommended by the Pointing Wizard led to an average of 29% fewer clicks per target, 22% faster target acquisition time, and a 27pp improvement in error-free targets (all significant at $P < 0.05$).

Object size

For object size, the larger targets in part 2 averaged 20% faster acquisition time as compared with the Windows default size targets (significant at $P < 0.05$). The effect ranged from a minimum of −2.71% to a maximum of 59%. This confirmed what we had learned from our earlier study: that overall a larger target works better, but not everybody receives significant benefits. The real question is whether our algorithm could recommend larger targets for those who would benefit, and not recommend them when not necessary.

To address this question, we used these data to analyze the specificity of our object size recommendations. Based on the first A condition in part 1, each subject received a recommendation of whether to use a larger target size. For the 11 people who completed all parts of the study, we compared their object size recommendation to whether larger objects actually enhanced performance time by 15% or more. Our original algorithm matched the actual results in only 4 of 11 cases. We identified three ways of improving the algorithm: to use only single-click trials, to be less aggressive in filtering outliers, and to consider selection success in addition to selection time. Revising the algorithm accordingly led to an 8 of 11 success rate, with a sensitivity of 100% and a specificity of 50%. This revised algorithm was implemented into the Pointing Wizard.

Discussion of effectiveness studies

In these studies, our algorithms generally did a good job of recommending new settings for users who needed them and not recommending them for users who did not. The sensitivity measures for the StickyKeys, pointer speed, and object size algorithms were all 100%. The specificity measures were above 80% for StickyKeys and pointer speed, and 50% for object size. So there were a few false-positive recommendations, but no false negatives in our studies to date. Participants generally fell into three categories: (1) those whose performance benefitted from the algorithms' recommendation of a non-default setting; (2) those whose performance seemed unaffected by the algorithms' recommendation of a non-default setting; and (3) those for whom the algorithms recommended continued use of the Windows default settings. Category 2 was observed almost exclusively for the pointer speed setting, in which a recommended pointer speed of 12, say, would result in very similar performance to the default pointer speed of 10. Note that there were no subjects who performed notably worse with the settings recommended by our algorithms.

For those who do need changes in their Windows settings, our algorithms recommended settings that led to improved performance. Table 4 summarizes the results across our effectiveness studies.

While all of the participants have physical impairments that affect their ability to use a keyboard or mouse, not all of them required an adjustment in their Windows settings. The double-click settings benefitted the largest percentage of subjects and provided consistently enhanced performance. The repeat settings benefitted fewer individuals, but had the most dramatic impact for those who needed customized settings. Pointer speed may be the setting that had the least consistent affect on user performance, but even there, adjustments did significantly benefit some individuals.

Usability studies

A major goal was to ensure that our software wizards were easily usable by end users, regardless of their physical impairment or their prior familiarity with keyboard and mouse settings. To ensure that this goal was met, we

Table 4 Summary of effectiveness studies

Setting	Subjects who benefitted (%)	Speed increase (%)	Accuracy increase (%)
Repeat rate and repeat delay	17% (2/12)	36	96% Fewer errors
StickyKeys	50% (6/12)	14	100% Fewer errors
Pointer speed	25% (3/12)	9 to 23	>40% Fewer entries
Double-click time and double-click distance	~80% (19/24)	20	~30% Fewer clicks
Object size	45% (5/11)	19 to 59	17 to 33% Fewer entries

Subjects who benefitted is the % of subjects who received a notable performance benefit from using the settings recommended by our algorithms.

performed six usability studies across the 2-year project.[34] The approach and results are summarized below.

Initial user interface design

A series of interviews with practitioners and end users helped us define the initial feature set for our software and establish some early usability criteria. Based on these initial ideas, we developed a wireframe prototype system, using Axure RP [Axure, Inc.] software. We decided to focus on Keyboard Wizard development first, to keep a manageable scope. The wireframe prototype was a clickable mockup of the system, showing what each screen and transition would look like. However, it did not actually make real adjustments to the keyboard settings; it merely acted as if it did.

Wireframe usability studies

Methods

Across two studies, 14 individuals went through a usability protocol with a wireframe interface. Six participants were end users, with a variety of physical impairments; five were assistive technology practitioners; and three were caregivers or friends of an end user. The protocol occurred over a single session, and included the following steps:

1. 14 background questions about the participant and their computer use;
2. A basic scenario that asks the participant to walk through the Keyboard Wizard interface, with no guidance from the experimenter;
3. Open-ended questions regarding specific areas of the interface for which we needed more user input;
4. Likert-type questionnaire items regarding ease-of-use and other aspects of the interface.

The sessions were video-recorded using Morae Recorder [Techsmith Corp.] software to capture the screen and all user actions and comments. This allowed detailed review of usability issues experienced by the participants and provided a way to determine the time it took each user to complete the scenario task.

Specific variables that were quantitatively analyzed were completion of scenario task (yes/no), completion time (minutes), and usability ratings (1 to 5). However, qualitative observations of usability problems and participant comments were more important at this stage.

Results

Questionnaire responses led to a benchmark completion time of 15 minutes or less. Actual completion times ranged from 4 to 16 minutes, averaging 8.1 minutes. All participants were able to complete the task without significant difficulty. End users rated the ease-of-use at 3.7 on a scale of 1 (low) to 5 (high).

While these results looked reasonably promising, the fact that the wireframe prototype did not actually change the user's settings was a significant limitation. The main usability issue related to understanding when and whether the wizard actually adjusted the Windows settings. Some users wondered whether they were supposed to make the recommended changes themselves. This confusion was due at least in part to poor interface design, but the limited functionality of the prototype certainly made the problem worse.

The primary revisions made were to clarify ambiguous wording, with some minor workflow adjustments. After revising the prototype through several rounds of changes, the results suggested that end users could use this interface successfully.

We implemented a fully working prototype of the Keyboard Wizard using the Java programming language. This addressed the limitation of the wireframe, since the working prototype actually did activate the Windows settings correctly for the user.

Usability studies for keyboard wizard prototypes

A series of four usability studies were conducted with four versions of the working prototype. For three of these studies, the methods were very similar to those used for the wireframe studies, using the same scenario task and questionnaire. Some of the open-ended questions varied between studies, depending on particular issues that we were exploring. The fourth study was a remote beta test, in which we made the beta version

available for download and asked users to complete an online survey. Key results are described below.

Results with version 0.0

This study involved four participants: three end users and one practitioner. Results were illuminating, as research notes refer to it as a "usability disaster" for one end user in particular. A major problem related to trouble with the Enter key, which activated the Next button on each screen. Holding the Enter key down too long moved through multiple screens unexpectedly, and hitting it inadvertently suddenly advanced to the next screen. This led to a large amount of confusion for this user, although she was eventually able to complete the task.

While the other two end users did not have that same difficulty with the Enter key, there were still 11 major issues identified. For example, participants preferred to choose whether to use the recommended settings on the basis of a brief practice rather than formal data collection. This led to a revised workflow in which users can opt-in to additional data collection if desired, but it is no longer on the default path. Additionally, the user interface was simplified to remove the sidepanel, which tended to create more confusion than it prevented.

Average completion time was 11 minutes for the end users, and ease-of-use rating averaged 3.33. These compare unfavorably to values observed in the wireframe studies, as we would expect, given the usability problems observed.

Despite the numerous issues, most of the solutions were straightforward. The biggest challenge was to resolve the keyboard issues so that inadvertent Enter hits would not cause problems, while still preserving 100% keyboard access for people who have difficulty using a mouse.

Results with versions 0.1 and 0.2

These two studies involved a total of seven end users and one practitioner and included the individual who had difficulties with the Enter key in the previous study. The problems with Enter were solved, but a second user revealed issues with inadvertent spacebar presses. As the spacebar could also act as a button click, under certain conditions, this led to unexpected and surprising screen transitions. A response time issue emerged, as some steps in the wizard had long delays when transitioning to the next step. These issues were addressed in version 0.2. All version 0.2 users completed the task with ease. Revisions at this point were refinements to screen text, additional enhancements to response time, and other relatively minor issues.

Across these two studies, average completion time was 5.5 minutes for end users. Ease-of-use rating averaged 4.83 (on a scale of 1 to 5).

Beta test with version 0.9

Given the high usability observed with version 0.2, the next step was to create a beta version for wider distribution. While we were still interested in usability at this point, a main goal of the beta test was to identify any glitches in program installation and execution across a wide range of computers and computing environments. Seventeen individuals participated, after responding to notices to the QIAT and RESNA listserv communities. Fourteen of these were practitioners, and three were end users with physical impairments. Participants were asked to walk through specific tasks with the wizard and complete a questionnaire. Unlike the previous studies, however, they were not observed or recorded while using the software. No significant functional or usability problems were reported. Ease-of-use rating averaged 4.4.

Usability study for pointing wizard

The Pointing Wizard is completely analogous to the Keyboard Wizard, so its user interface is quite similar. So, while development of the internal pointing algorithms continued during revisions for Keyboard Wizard, we did not implement a Pointing Wizard prototype until we were confident in the usability of the wizard interface. Once the Keyboard Wizard beta test was complete, we developed the Pointing Wizard to use a very similar user interface (Figs. 2 and 3). 1

Six end users participated in a usability study for Pointing Wizard. This followed the same approach as the Keyboard Wizard studies, using an analogous scenario task and questionnaire, and all sessions were recorded.

Several minor issues were identified and fixed, but overall everyone was able to complete the task easily. Completion time for running through the wizard averaged 6 minutes, and ease-of-use rating averaged 4.5.

Discussion of usability studies

By the completion of these usability studies, both Keyboard Wizard and Pointing Wizard were ready for general release, and we were confident in their high level of usability for end users as well as practitioners. Each type of study performed was useful in different ways. The wireframe prototypes provided an efficient way to explore overall look-and-feel and workflow

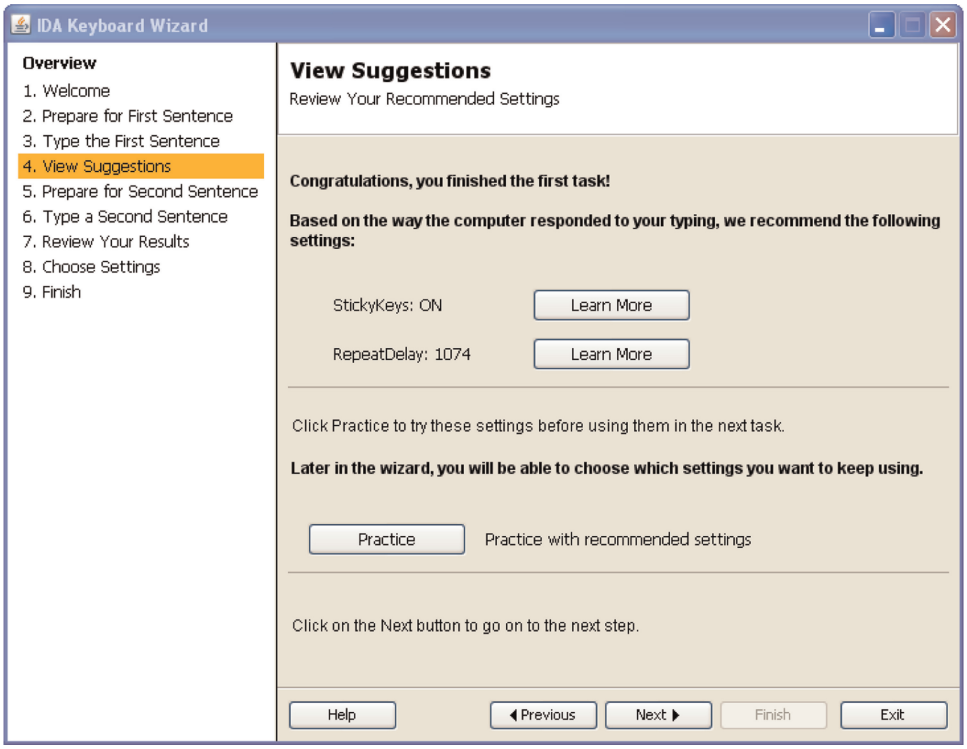

Figure 2 Screenshot from Keyboard Wizard v0.0, showing sidepanel, small font, and required second typing task. These usability problems were identified and corrected in subsequent versions.

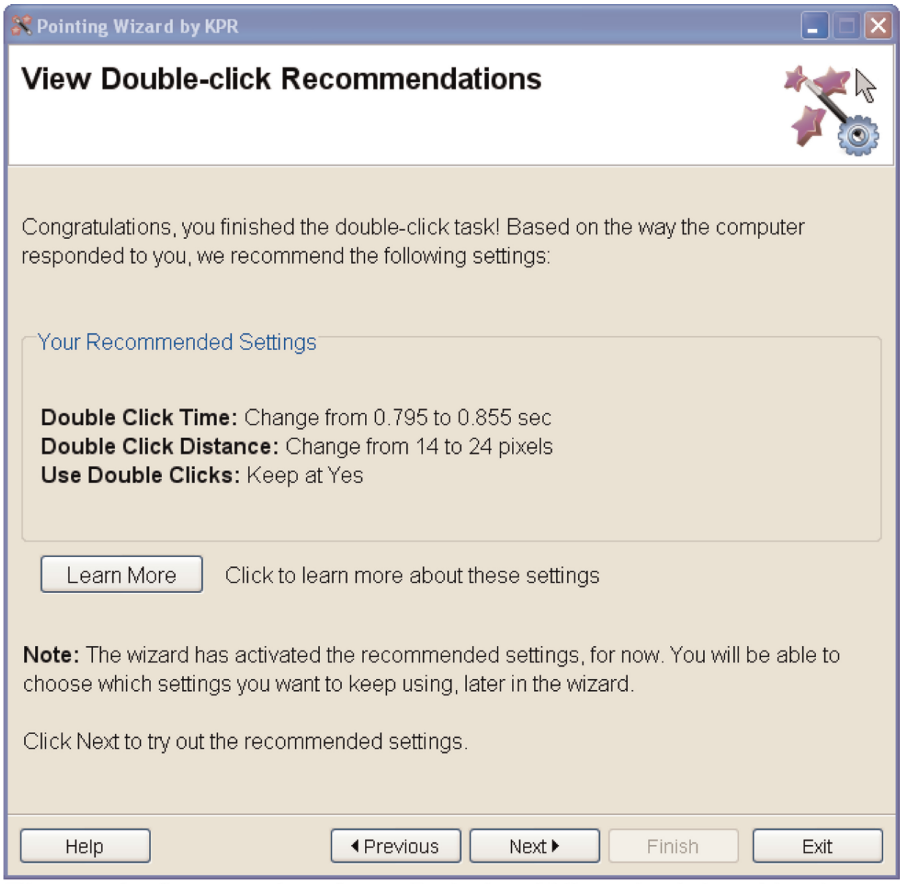

Figure 3 Screenshot from Pointing Wizard v1.0, with clearer organization, larger font.

issues. Due to their limited functionality, however, extensive testing was still required with fully functional prototypes. This is particularly true for this application in which it was impossible to make the wireframe fully mimic all aspects of the real system. Remote beta testing was a valuable final step to ensuring proper functioning across various computing environments with a larger number of users.

Discussion

The Keyboard and Pointing Wizards can be used by any end user who wants to optimize their keyboard and mouse settings. In most cases, the wizards recognize when changes are not needed, so they do not bother the user with unnecessary suggestions. When changes are recommended, the user has the final say over whether to activate those settings, and has the option to try the new settings prior to making a decision. Practitioners can also use the wizards in conjunction with an end user; reports provided by the wizards record the suggested settings and their outcomes for inclusion in clinical documentation or sharing with other interested stakeholders.

One concern is how well the performance measured within the wizard tasks generalizes to real-world performance during typical computer use. Our pointing studies directly addressed that concern by including some real-world (albeit scripted) tasks in the protocol. In general, the results on the scripted real-world tasks were consistent with those on the more constrained wizard tasks. In addition, construct validity for the Pointing Wizard test has been examined and shown to be high, with 83% of the variance in real-world task times explained by Pointing Wizard test performance.[35] This does not completely alleviate the generalizability concern, but it does provide confidence that the wizard test measurements have a direct relationship to performance on real-world tasks.

Future work includes adding more settings to the existing wizards, including SlowKeys, BounceKeys, and dragging support. We would also like to examine the effectiveness of the wizards for other types of users, such as older adults. Additionally, we are currently working on moving beyond a separate wizard task to using natural data as the basis for the system's recommendations. In other words, as the user goes about their typical computer tasks, the system monitors performance and suggests any changes in keyboard or mouse settings that may be warranted. This will save the user the few minutes required to run through the dedicated wizard tasks, and will also allow Windows to adapt more responsively to changes in a user's ability that may occur across time.

The Keyboard and Pointing Wizards were designed in the spirit of Wobbrock *et al.*'s[36] framework of ability-based design and particularly support its principles of ability, accountability, performance, adaptation, and transparency. Some practitioners have wondered why the Keyboard Wizard bothers to adjust repeat settings instead of just disabling the repeat feature altogether. Or similarly, why the Pointing Wizard adjusts the double-click settings rather than always recommending the single-click-to-open setting to avoid double-clicks. The reason is that this approach would violate the principle of *ability*, or taking advantage of all that users *can* do. In the repeat example, the user *can* control repeating keys, just with a different timing profile than Windows is configured for. If repeat is disabled altogether, it certainly will prevent further repeat errors from occurring. But it will also prevent the user from being able to use repeating keys intentionally, such as using the arrow keys to move around in documents, or using the backspace to remove multiple keystrokes. In the double-click scenario, the user *can* double-click; it simply takes them longer. If they prefer double-clicking, they should be supported in doing so. The Keyboard and Pointing Wizards try to help Windows leverage the user's abilities and accommodate their limitations, thereby supporting the user's choice in using their computer in the ways that they prefer.

Conclusions

We have developed two software wizards for Windows that help ensure that keyboard and pointing devices are properly configured for an individual, and reconfigured as the user's needs change. A series of effectiveness studies demonstrated that the wizards' recommendations can enhance productivity for computer users with physical impairments. Additionally, usability studies showed that most end users can readily use the wizards on their own. The wizards are a simple yet effective way of adjusting Windows to accommodate a user's physical impairments.

Acknowledgments

This work was supported by Grant Number R44HD045015 from the National Institute of Child Health And Human Development. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute of Child Health And Human Development or the National Institutes of Health. Thanks to Edmund LoPresti and Glen Ashlock for their

contributions to the software and data collection. We thank our participants for their generous contributions of time and effort.

References

1 National Council on Disability. Keeping track: national disability status and program performance indicators. Washington DC: National Council on Disability; 2008.
2 Hedrick B, Paper T, Heinemann A, Ruddell J, Reis J. Employment issues and assistive technology use for persons with spinal cord injury. J Rehabil Res Dev 2006;43(2):185–98.
3 National Institute of Neurological Disorders and Stroke. Multiple sclerosis: hope through research [document on the Internet] 2010. [retrieved 24 August 2010]. Available from <http://www.ninds.nih.gov/disorders/multiple_sclerosis/detail_multiple_sclerosis.htm>.
4 ALSMND Alliance. What is ALS/MND? [document on the Internet] 2010. [retrieved 24 August 2010]. Available from: <http://www.alsmndalliance.org/whatis.html>.
5 Bjelland M, Burkhauser R, von Schrader A, Houtenville A. 2009 Progress report on the economic well-being of working-age people with disabilities. Technical Report. Rehabilitation research and training center on employment policy for persons with disabilities; 2009.
6 Dobransky K, Hargittai E. The disability divide in Internet access and use. Inf Comm Society 2006;9:313–34.
7 Stevenson B, McQuivey J. The wide range of abilities and its impact on computer technology. A research study commissioned by Microsoft Corporation and conducted by Forrester Research, Inc.; 2003. Available from: http://www.microsoft.com/enable/research/default.aspx.
8 National Council on Disability. Empowerment for americans with disabilities: breaking barriers to careers and full employment. Washington DC: National Council on Disability; 2007.
9 Bureau of Labor Statistics. Overview of the 2008–18 projections. In Occupational Outlook Handbook 2010–2011 Edition. Available from: http://www.bls.gov/oco/oco2003.htm.
10 Anson D. Alternative computer access: a guide to selection. Philadelphia, PA: F.A. Davis Company; 1997.
11 Chen C-L, Chen H-C, Cheng P-T, Chen C-Y, Chen H-C, Chou S-W. Enhancement of operational efficiencies for people with high cervical spinal cord injuries using a flexible integrated pointing device apparatus. Archi Phys Med Rehabil 2006;87(6):866–73.
12 Drainoni M, Houlihan B, Williams S, Vedrani M, Esch D, Lee-Hood E, *et al.* Patterns of Internet use by persons with spinal cord injuries and relationship to health-related quality of life. Arch Phys Med Rehabil 2004;85(11):1872–9.
13 Trewin S, Pain H. A model of keyboard configuration requirements. Behav Inf Technol 1999;18(1):27–35.
14 Kantner L, Rosenbaum S. Usable computers for the elderly: applying coaching experiences. IPCC 2003 Proc IEEE Professional Communication Society; 2003.
15 Trewin S, Pain H. Keyboard and mouse errors due to motor disabilities. Int J Human-Computer Studies 1999;50:109–144.
16 Koester HH, LoPresti E, Mankowski J, Simpson R, Mankowski R, Ashlock G. Automatically adjusting pointing device settings enhances user double-click performance. In Emiliani PL, (ed.) Assistive technology from adapted equipment to inclusive environments. Amsterdam: IOS Press; 2009.
17 Koester HH, LoPresti E, Simpson R. Toward automatic adjustment of keyboard settings for people with physical impairments. Disabil Rehabil: Assist Technol 2007;2(5):261–74.
18 Stevenson B, McQuivey J. Examining awareness, use, and future potential. A research study commissioned by Microsoft Corporation and conducted by Forrester Research, Inc.; 2003. Available from: http://www.microsoft.com/enable/research/default.aspx.
19 Goodman N, Jette AM, Houlihan B, Williams S. Computer and internet use by persons after traumatic spinal cord injury. Arch Phys Med Rehabil 2008;89(8):1492–8.
20 McGill R. A blackboard knowledge-based approach towards implementing an adaptive force joystick computer input device for persons with tremor disability. Proceedings of RESNA 1990 Annual Conference; 1990. p. 431–2.
21 Tracey M, Winters J. Neuro-fuzzy advisor for mouse setting in Microsoft Windows. Proceedings of the BMES-EMBS 1st Joint Conference; 1999. p. 664.
22 Simpson R, Koester H. Adaptive one-switch row-column scanning. IEEE Trans Rehabil Eng 1999;7(4):464–473.
23 Lesher G, Higginbotham DJ, Moulton B. Techniques for automatically updating scanning delays. Proceedings of the RESNA 2000 Annual Conference. Arlington, VA: RESNA Press; 2000. p. 129–31.
24 Trewin S. Automating accessibility: the dynamic keyguard. Proceedings of ASSETS 2004. New York: ACM. 2004. p. 71–8.
25 LoPresti EF, Brienza DM. Adaptive software for head-operated computer controls. IEEE Trans Rehabil Eng 2004;12(1):102–11.
26 Hurst A, Hudson S, Mankoff J. Automatically detecting pointing performance. IUI'08, New York: ACM. 2
27 Trewin S, Pain H. Dynamic modelling of keyboard skills: supporting users with motor disabilities. Proceedings of the 6th International Conference on User Modeling. Wien, NY: Springer; 1997. p. 135–46.
28 Trewin S. Configuration agents, control and privacy. Proceedings of Conference on Universal Usability 2000. New York: ACM; 2000. p. 9–16.
29 Trewin S. An invisible keyguard. Proceedings of ASSETS 2002. New York: ACM; 2002. 143–9.
30 Dawe M. Complexity, cost and customization: uncovering barriers to adoption of assistive technology. Refereed Poster at the ACM SIGACCESS Conference on Computers and Accessibility. Atlanta, GA; 2004.
31 Koester HH, LoPresti E, Simpson R. Factors influencing user performance with pointing devices. Proceedings of RESNA 2006 Annual Conference, Atlanta, GA. Arlington, VA: RESNA Press; 2006.
32 LoPresti E, Koester HH, Simpson R. Toward automatic adjustment of pointing device configuration to accommodate physical impairment. Disabil Rehabil Assist Technol 2008;3(4):221–35.
33 Koester H, LoPresti EF, Simpson RC. Toward Goldilocks' pointing device: determining a "just right" gain setting for users with physical impairments. Proceedings of the 2005 ASSETS Conference. New York: ACM Press; 2005.
34 Koester H, Mankowski J, LoPresti E, Ashlock G, Simpson R. Software wizards for keyboard and mouse settings: usability for end users. Proceedings of RESNA 2011 Annual Conference, Toronto, ON, Canada. Arlington, VA: RESNA Press; 2011.
35 Koester HH, LoPresti EF, Simpson R. Further validation of the "Aim" Test for assessing a user's mouse skills. Proceedings of RESNA 2011 Annual Conference, Toronto, ON, Canada. Arlington, VA: RESNA Press; 2011.
36 Wobbrock J, Kane S, Gajos K, Harda A, Froehock J. Ability-based design: Concept, principles and examples. ACM Transactions on Accessible Computing 2011;3(3):9:1–9:27.

Authors Queries

Journal: **The Journal of Spinal Cord Medicine**
Paper: **JSCM-D-12-00068**
Article title: **Software wizards to adjust keyboard and mouse settings for people with physical impairments**

Dear Author
During the preparation of your manuscript for publication, the questions listed below have arisen. Please attend to these matters and return this form with your proof. Many thanks for your assistance

Query Reference	Query	Remarks
1	Please confirm the given citation of Figs 2 and 3, else provide the appropriate text citation.	
2	Please provide publication year in ref. 26.	