MAKING CUSTOM KEYBOARD AND MOUSE SETTINGS PORTABLE

Heidi Koester, Jennifer Mankowski
Koester Performance Research, Ann Arbor MI

INTRODUCTION

In previous work, we developed two software wizards used to customize keyboard and pointing device settings to optimize user productivity and comfort [1]. Recently we added a new feature to both of the wizards which allows packages of Windows registry settings to be activated on other computers. This paper describes the new portable settings feature, reasons why it is desirable, and the manner we chose to support it.

BACKGROUND

The Windows operating system supports a variety of registry settings that control keyboard and pointing device behavior. Determining the optimal values for these registry settings can be somewhat complex and confusing for the typical computer user. Our software wizards simplify the process of determining the optimal setting values for an individual and activating those settings.

Keyboard Wizard presents a text entry task in which the user is asked to transcribe a short sentence. At completion of the task, the wizard analyzes the user's text entry data and makes recommendations for Sticky Keys, keyboard repeat delay and repeat rate. The user can practice with the recommended settings prior to selecting which settings they would like to keep using. At any point throughout the wizard, the user can choose to return to their original settings and exit the application.

Pointing Wizard consists of two parts which walk the user through the process of optimizing basic target selection settings and double-click settings. Each part follows a similar flow to that described for Keyboard Wizard. Part One makes recommendations for object size, pointer speed, and Enhance Pointer Precision. Part Two makes recommendations for double-click time, double-click distance, and whether double-clicks should be used at all.

The first version of the wizards had a significant shortfall in that there was no provision for storing the settings packages in a form that could be used to easily activate the settings on any Windows machine. This meant that a user would need to run through the wizard on each machine they used, or manually transfer the settings via the Control Panel. Either option is inconvenient and time-consuming. It also meant that for machines which are shared among multiple users without individual accounts, it would be difficult to switch the settings back to the originals and then re-activate the custom settings when desired.

To overcome these limitations, we wanted to allow users to easily activate the wizard's final settings on any computer. We determined that restoring Windows default settings could be a desirable function as well, since it would allow for a 'reset' if there were multiple users of a machine or if the user had a change of heart regarding which settings they wanted to use. Providing the settings in a portable, easily activated form would allow clients to run through the wizards in the clinic or school and then take the settings home or to public computer labs for use on other machines.

REQUIREMENTS

Our primary goal was to provide the ability for users to activate settings packages created by the wizards on other machines with minimal changes to the wizard user interface. In that spirit, we considered the new feature as two distinct parts consisting of the wizard's responsibility for interfacing with the user and creating the packages and the resulting packages themselves. The following requirements emerged.

Wizard

W1. The user will be presented with the option of saving the active settings package on the wizard Finish screen. This will eliminate confusion as to what setting values will be included in the package.

W2. The wizard will allow the user to specify a filename and location for the package.

W3. The wizard will write a settings package file with the currently active settings to a file.

W4. The wizard will write an additional settings package to the same location. This package will contain the Windows default settings applicable to the wizard, keyboard settings for Keyboard Wizard and pointing related settings for Pointing Wizard. This default settings file will allow the user to undo changes made by activating the wizard settings package.

Settings Packages

S1. Each portable settings package will consist of a single file. This supports portability in the simplest sense because the user will only need to move or copy a single file.

S2. The portable settings package should be easy to use. The user should understand what to do with the file intuitively.

S3. Each settings package file will be small, less than 100 kB, to make file transfer trivial.

S4. Execution of the file will write the desired setting values to the Windows registry. Ideally it will also activate the settings without requiring any additional actions by the user.

S5. When executed, the file will read in certain registry values such as MenuFont and ShellState and perform operations to modify specific bytes. This is necessary because the entire fields are complex and control much more than what we intend to change. If the read operations are unsuccessful, nothing will be written to those values.

S6. The packages will support all settings recommended by Keyboard Wizard and Pointing Wizard.

DESIGN

Wizard

The requirements for the additions to the wizard were relatively straight forward: adding a new button and a file chooser in addition to creating the settings package files. The main challenge was explaining the feature clearly and concisely so that the user understands how to use the package files. This is a requirement of the settings packages themselves, but because they are stand-alone files, it may be difficult to embed information on how to use the file. It makes sense to take advantage of the opportunity that we have within the wizard to explain the use of the files by displaying a dialog box after the files have been written.

Settings Packages

The primary challenge here was determining what type of file to use for the portable settings packages. Based on the comparison of available registry scripting methods [2], the file types that we considered were REG, INF, batch files, and VBScript.

Registry files (.REG extension) are system files that update the Windows registry when run. This is a standard method provided by Microsoft for modifying the registry. The user imports setting values by double-clicking on the file or right-clicking and selecting ‘Run’. Regarding requirement S2, even though REG files are an accepted way to change the registry, it is unclear how many of our users would recognize the file type and understand how to use it. After the file has been executed, the user must log off and log back on or restart the computer to activate the new settings. This presents a significant usability issue because there is no way to inform the user that further action is required to activate the settings. Additionally REG files do not satisfy requirement S5 because they cannot query values.

Information files (.INF extension) are installer files typically used to install applications or hardware. To write the setting values, the user would right-click the file and select ‘Install’. INF files could be a bit misleading in this context since nothing is really being installed. They present the same usability issues as REG files in the sense that use of the file is not intuitive and there is no way to inform the user of the need to restart the machine to activate the new settings.

Batch files (.bat extension) are used to execute commands with the Windows Command Prompt. Using a batch file, we can run REG.EXE (a console registry tool provided with Windows) to read from and write to the registry. Loops and various forms of variable manipulation are also supported and information can be written to the command window informing the user of the

need to log off and log back on or restart the machine to activate the new registry settings.

None of the file types that we examined completely satisfied requirement S4 because they all require further action to activate the newly written registry settings. This seems to be unavoidable when using a single-file solution since activation of the settings that are being changed is problematic. We have found that standard methods of notifying all applications to update their SystemParametersInfo are not effective. We accept this limitation, but it does create an additional requirement to inform the user to log off and log back on or to restart the computer to activate the new settings.

We decided to use batch files for our portable settings package because they provide the majority of the functionality specified in our requirements, with the possible exception of intuitive use (S2). It is unlikely that our users will recognize the batch file type and immediately know how to use it. However, directions can be provided both in the wizard and in the wizard help system. Additionally, when a person does not know what to do with a file, typically they double-click on it or try to run it in some fashion, which is exactly the correct action for running batch files.

VBScript is an option that was explored after we had already shown the feasibility of the batch file approach. VBScript would provide a GUI to our users but it would still not activate the settings without requiring a restart. Because the batch file approach is simpler and VBScript would not provide increased functionality, we decided to use batch files.

IMPLEMENTATION

Several additions were made to the wizard interface. These include a 'Save Batch File' button on the Finish screen offering the user the option to store the active settings to a file, a file chooser dialog displayed when the button is activated, and an instructional dialog box displayed after the file has been created.

Figure 1 shows the instructional dialog box displayed from Keyboard Wizard when a settings package file has been successfully saved.

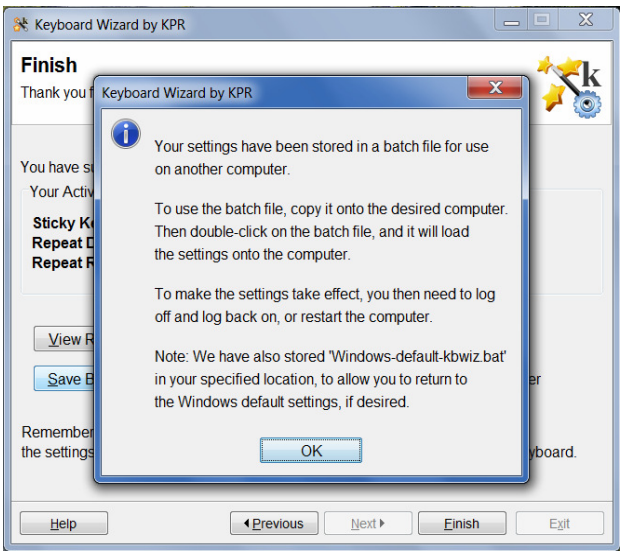

Figure 1. Instructional dialog box displayed by Keyboard Wizard after settings have been saved to a batch file.

Figure 2 shows a wizard-created batch file as seen from Windows Explorer.

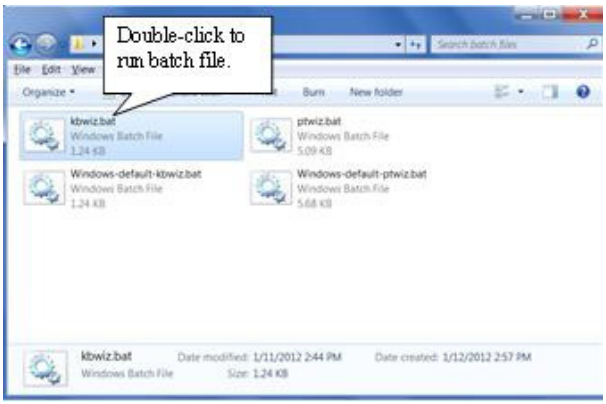

Figure 2. Batch file as seen from Windows Explorer.

When the user double-clicks on the batch file or right-clicks and selects 'Run', a command prompt window opens and the commands from the batch file scroll through that window as they are executed. Figure 3 shows the command prompt window when execution of the batch file has completed.

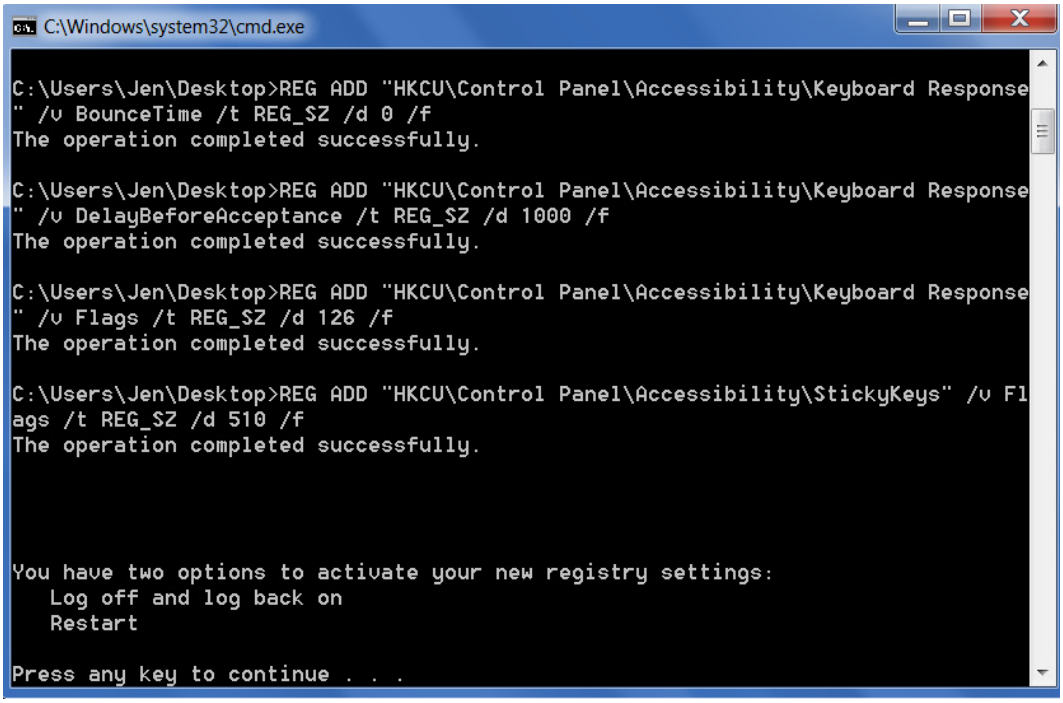

Figure 3. Window at completion of batch file execution.

CONCLUSION

We have successfully added the portable settings feature to Keyboard Wizard 1.2 and Pointing Wizard 1.2. Wizard requirements W1 through W4 and settings package requirements S1, S3, S5, and S6 have been satisfied completely. With the instructions added to the wizard interface, requirement S2 has been met. We believe that users will understand or at least be able to guess how to run the batch file. Requirement S4 has been met to the greatest extent possible while still supporting a single-file solution. Additional instructions provided from the batch file clarify the need for a restart to activate the new settings. Ideally we will incorporate usability testing specific to this feature in some of our future studies to get user feedback on the design. Batch file behavior has been confirmed on Windows XP and Windows 7 from both administrator and standard user accounts.

This feature gives wizard users the ability to store packages of registry settings in batch files. This batch file provides portability so that individuals can activate settings to customize keyboard or pointing device behavior on other machines without running either Keyboard Wizard or Pointing Wizard.

ACKNOWLEDGEMENTS

This work was supported by the Paralyzed Veterans of America Research Foundation and the U.S Department of Education (NIDRR).

REFERENCES

[1] Koester, H.H., Mankowski, J., LoPresti, E.F., Ashlock, G., Simpson, R. “Software Wizards for Keyboard and Mouse Settings: Usability for End Users”, in Proceedings of RESNA 2011 Conference, Toronto, Canada: RESNA Press, 2011.

[2] Honeycutt, J. (2005). Microsoft Windows Registry Guide, Second Edition. Redmond, Washington: Microsoft Press.