Seminar Report

AutoHotkey

Ursula Reiterer

5 February 2013

Supervisor: Michael Schaper

Abstract

AutoHotkey is an open-source software utility for the automation of tasks on Microsoft Windows systems, which is meant to be usable by programmers as well as non-programmers. It has been developed since 2003 as an autonomous language by Chris Mallet, until he officially quit working on the project in 2010.

AutoHotkey was derived from the automation language AutoIt v2, but was extended to cover additional features like hotkeys or hotstrings. Small macros can be written to assign different events to combinations of keystrokes or mouse clicks. Furthermore, scripts can be compiled and distributed to computers running Windows.

Contents

1 Introduction

AutoHotkey is a free, open-source utility for Windows, which enables the user to automate repetitive tasks. For example, by using hotkeys for keyboard and mouse, or so-called hotstrings to facilitate writing. It was designed as an easy-to-learn scripting language and is still used by programmers as well as non-programmers.

This article gives a short overview of AutoHotkey. In Section 2, we discuss the history of the language, the events that led to its development, and possible applications. In Section 3, we summarize the features provided by the language and present its basic usage. At this point, we are ready to discuss the AutoHotkey implementation of the '99 Bottles of Beer' song. To round up the overview, we look briefly at characteristics of AutoHotkey, differences to well-known languages and alternatives to it in Section 4.

2 History and Application

In this section we give a brief history and describe typical applications of AutoHotkey.

2.1 History

AutoHotkey was developed by programmer Chris Mallet [1]. He released a first beta version of the language on November 10, 2003 and the final version 1.0 on February 16, 2004 [2].

Mallet first offered to integrate a hotkey support into AutoIt v2, a freeware macro language for Microsoft Windows [3]. However, the AutoIt community was not supportive, and Mallet decided to develop AutoHotkey as a standalone tool, strongly based on AutoIt and its compiler.

This incident most probably led the main developer of AutoIt, Jonathan Bennett, to switch from GPL (GNU General Public License) to closed source soon afterwards. AutoHotkey however remains under GPL.

> *"Before I start this topic, this is NOT going to turn into a flame topic and no product names will be mentioned. [...]*
> *I've just about had it with numerous other projects repeatedly taking AutoIt code – against the expressed wishes of the developers – including it whole in their own code and then setting themselves up as competitors. [...] When we complain we generally get the response 'it's GPL so tough'"* [4]

On October 10, 2010 Mallet announced in the AutoHotkey forum, that – although he already had plans for AutoHotkey v2 – he had lost interest in the project and would therefore not actively contribute to its further development.

> *"My original motivation for starting the AutoHotkey project was to empower users, including non-programmers, to do automation*

and hotkeys. I wanted the scripting language to be kept simple yet flexible. I didn't want it to become a full-blown programming language." [5]

In the same post, he endorsed AutoHotkey_L, an extension of AutoHotkey developed by members of the community. AutoHotkey_L is now the official development branch, with support for arrays, objects and other features [1].

The present article focuses on AutoHotkey v1, not on the AutoHotkey_L development.

2.2 Typical Applications

AutoHotkey is a free, open-source tool. The features are limited and only a small number of commands is supported, in an attempt to make the language easy to learn. It can be used to simplify tasks, for example by remapping keys on the keyboard or launching multiple programs by a single click. Hotstrings can be created, that replace specified abbreviations with longer sentences to facilitate writing. However, these features are primarily intended to create small tools for the individual user. It is not used for bigger tasks, and is certainly not a development language.

3 Usage

We discuss features and the basic usage of the language. At the end of the section, we take a closer look at the '99 Bottles of Beer' song and application examples of AutoHotkey.

3.1 Features

We give a short overview of the features of AutoHotkey. We discuss the utility in general, not only the language.

Mouse and keyboard macros

AutoHotkey enables the user to write macros to speed up repetitive tasks.

Macros are sequences of commands, that are combined to be executed in a simpler way. Usually these sequences are needed several times by a user, such that setting up a macro is time saving and less vulnerable to errors than repeating the same procedure itself.

For example, the preparation of a standard e-mail template: the macro could include the activation of Microsoft Outlook and the setup of a blank e-mail with subject title and sample content, which then can be edited. Within a self-written AutoHotkey script all these instructions could be connected to a certain input event, for example a hotkey. If the hotkey is used, all instructions are executed automatically in the predefined order.

Macro recorder

AutoHotkey provides a macro recorder named ‘AutoScriptWriter’, which can be used to record mouse clicks or keyboard entries, or to track active windows, without the need to write scripts. These actions can be converted into runnable AutoHotkey scripts, to be played-back later on.[1]

Compatibility with AutoIt v2 scripts

AutoIt v2 scripts can be edited and run inside AutoHotkey with only few exceptions. This is due to the fact that AutoHotkey is derived from AutoIt and the languages only slightly differ in the basic command set.

Executable files

AutoHotkey scripts can easily be converted into standalone executables files by running the compiler on an .ahk script. This way, scripts can be used on other computers without the need to install AutoHotkey.

3.2 Creating and Running Scripts

After the installation of AutoHotkey first scripts can be written. An AutoHotkey script is usually saved as .ahk. After double-clicking on it, it runs in background and is accessible via an icon in the Windows notification taskbar (another way to run it would be to compile it into an .exe file and launch it the usual way). A right click on the same icon displays different options, among them exiting the file. Multiple scripts can run simultaneously, showing an individual icon for each script.

3.3 Introduction to the Language

At this point we give a short introduction to the AutoHotkey language. A more detailed tutorial and the full documentation can be found on the official website [1].

Variables

In AutoHotkey, no data types have to be defined when using variables. The ‘=’ operator is used to store values in the variables, as shown in Listing 1.

```
Var1 = 123
Var2 = my string
```

Listing 1: Defining variables with the equal operator.

[1]Also if the AutoScriptWriter is explicitly mentioned on the website, it is not part of the actual AutoHotkey_L package anymore (now the main download on the website). In order to get access to the macro recorder it is necessary to download the ‘AutoHotkey Basic’ package separately, which has the recorder still included.

Variables are global within the script, except for local variables inside functions.

When referring to the content of a variable, the '%' has to enclose the variable name. For example, the code in Listing 2 would store '123 my string' in the variable 'VarConcatenated', which is then shown in a message box.

```
VarConcatenated = %Var1% %Var2%
MsgBox %VarConcatenated%
```

Listing 2: Re-use of variables.

The contents of variables can be compared with other variables or values by using certain operators (for example '>=').

Aside from the '=' sign, also the ':=' operator can be used to store values in variables. Mallet describes the latter as 'expression' method, and the other one as 'traditional' method [6], not further explaining why he integrated them both in AutoHotkey. The use of the expression operator is however slightly different, as can be seen in the lines of Listing 3, which are functionally identical to the ones in Listing 1 and Listing 2. Here, the later re-use of variables is made without '%' signs, and in the message box, a single '%' before the variable name is used to indicate that the content of the variable should be shown.

```
Var1 := 123
Var2 := " my string"
VarConcatenated := Var1 Var2
MsgBox % VarConcatenated
```

Listing 3: Defining variables with the colon-equal operator.

Unlike the '=' operator, ':=' can also be used, when a result of an expression, for example a mathematical operation, has to be stored in a variable, as in Listing 4.

```
NetPrice := Price * (1 - Discount/100)
```

Listing 4: A mathematical expression.

AutoHotkey also has a set of key variables like 'clipboard', which contains the current context of the Windows clipboard. They can be referenced by any script in the same manner as usual variables.

Conditions

Similar to other languages, AutoHotkey gives the possibility to use conditional constructs. A simple if-else statement can be seen in Listing 5. Depending on the value contained in the 'Color' variable, a message box with different answer is displayed.

```
if (Color = "Blue" or Color = "White")
{
    MsgBox It is blue or white.
}
else
{
    MsgBox This color is not recognized.
}
```

Listing 5: An if-else construct.

Loops

An example for a loop can be seen in the code of Listing 6, that will start the same message box five times. Instead of a variable, also a concrete integer could be written behind the '`Loop`' command to define the maximum number of iterations.

```
RunCount = 5
Loop %RunCount%
{
    MsgBox This window is displayed %RunCount% times.
}
```

Listing 6: A loop.

Besides the '`Loop`' command, also '`while`' is known in AutoHotkey, which can be used similar as in other languages.

Functions

Functions are defined by a function name, followed by parameters in brackets. To assign the return value of a function to a variable, the '`:=`' operator can be used together with the function call, as illustrated in Listing 7. A function can also call itself recursively [7].

```
Add(x, y)
{
    return x + y
}

Var := Add(2, 3)
```

Listing 7: A function.

Hotkeys

Commands can be assigned to hotkeys, to automatically execute them when pressing a key or mouse button. For example, a hotkey, which opens the Notepad when pressing `‘Win+n’`, could be created by a single line as in Listing 8. The `‘::’` operator defines the preceding combination as a hotkey, and the `‘#’` symbol is indicative for the Windows key. A list of symbols and operators used in AutoHotkey can be found on the website [1].

```
#n::Run Notepad
```

Listing 8: Running Notepad by a hotkey.

If more than one process should be initialized by a hotkey, the `‘return’` command can be used to signalize the end of instructions as illustrated in Listing 9. As can be seen here, the `‘Run’` command is also used to launch programs and documents under certain paths. By calling an URL, the website is opened within the standard browser of the user. Similar to that, an empty e-mail addressed to the specified person is opened, when using the `‘mailto:’` token.

```
#n::
  Run Notepad
  Run C:\My Documents\Address List.doc
  Run www.google.com
  Run mailto:someone@somedomain.com
return
```

Listing 9: Different events initiated by the same hotkey.

Hotstrings

AutoHotkey integrates so-called ‘hotstrings’; when writing a certain abbreviation, it is expanded to the full string, if so defined in the script. An abbreviation as `‘btw’` could for example be automatically expanded to the full string `‘by the way’`, when running a script as shown in Listing 10. Here, the abbreviation has to be included between two `‘::’` operators to define it as a hotstring.

Hotstrings work system wide for editors, file managers or browsers.

```
::btw:: by the way
```

Listing 10: A hotstring example.

Remapping keys and buttons

It is possible to remap keys or buttons with AutoHotkey. A script as shown in Listing 11 would for example map the `'y'` key to `'z'` and vice versa (this also includes the uppercase letters, such that `'Z'` is typed when using `'Shift+y'`). The same procedure can be used to remap mouse buttons, or also to disable keys or buttons if not needed (by assigning them to `'return'`).

```
y::z
z::y
```

Listing 11: A key remap example.

Sending keystrokes

A predefined text can be sent to the foremost window by the `'Send'` command. An example is shown in Listing 12, where the `'Control+Alt+s'` hotkey is chosen to transmit a standard input to an empty e-mail message. All characters are sent literally except {Enter}, which simulates the keystroke to begin a new line. There are more special characters and keys, including for example `'Control+c'` or `'Control+v'` strokes to copy or paste objects. The full list is available on the website [8].

```
^!s:: Send Sincerely,{Enter}John Smith
```

Listing 12: A 'Send' example.

Sending mouse clicks

To send a mouse click to a window by an AutoHotkey script, it is necessary to admit the X and Y coordinates of the desired position (relative to the active window). This can be done by the 'Window Spy', which is a small tool of AutoHotkey, that tracks mouse movements when activated. It can be launched from the Start Menu or tray-icon of a running script. The exact coordinates have then to be put behind a `'Click'` command as shown in Listing 13.

```
Click 112, 223
```

Listing 13: A 'Click' example.

Other mouse-specific commands are provided, for example to move the mouse to a position with an user-defined speed, or to click on a mouse button and to release it after a move.

GUI

AutoHotkey offers a set of GUI options. For example, message boxes of different types can be used to interact with a user as shown in Listing 14. The code starts a message box of type ‘4’ (a “Yes/No” button box) with a question aimed to the user. If ‘No’ is clicked, the script simply returns, otherwise a new message box confirms the user pressed ‘Yes’.

```
MsgBox, 4, , Would you like to continue?
IfMsgBox, No
    return
MsgBox You pressed YES.
```
Listing 14: A message box example.

A more complex sequence of GUI instructions can be seen in Listing 15. The first five lines of this code are used to set up the GUI layout and options. When running the script, the window ‘Example’ will appear, with the request to enter the name in the text-field. The last five lines close the window after the ‘OK’ button has been hit by the user, submit the entry to the variable and open a message box with the name.

```
Gui, Add, Text,, Enter your name:
Gui, Add, Edit, vName
Gui, Add, Button, default, OK
Gui, Show,, Example
return

GuiClose:
ButtonOK:
Gui, Submit
MsgBox Your name is %Name%.
ExitApp
```
Listing 15: A GUI example.

This is only a small example of GUI application in AutoHotkey. For further information see the ‘GUI’ chapter on the official website [9].

Manipulating windows

There is a set of commands for window manipulation in AutoHotkey; among them ‘`WinActivate`’ to make a window the foremost, or ‘`IfWinExist`’ or ‘`WinWait`’ to ask or control the status of a window, as illustrated in Listing 16. The code is activating a window titled ‘Untitled - Notepad’ in the Notepad, if it is already open, or otherwise launches Notepad, waits until the window with the defined title appears, and then activates it.

Manipulating files

Different commands exist in AutoHotkey to use or manipulate files. For example, the command ‘`FileAppend`’ can be used to add new lines at the end of a

```
IfWinExist Untitled - Notepad
{
    WinActivate
}
else
{
    Run Notepad
    WinWait Untitled - Notepad
    WinActivate
}
```

Listing 16: Window-specific instructions.

text file as in Listing 17. Other examples are instructions like `‘FileDelete’`, `‘FileCopy’` or `‘FileRead’` (into a variable), which are self-explanatory.

```
FileAppend, HelloWorld`n, File.txt
```

Listing 17: Appending a text to a file.

3.4 99 Bottles of Beer

By now, it should be possible to understand the ‘99 Bottles of Beer’ song shown in Listing 18.

```
b = 99

Loop, %b% {
        s .= b . " bottles of beer on the wall,`n"
        . b . " bottles of beer.`n
        Take one down, pass it around,`n"
        . --b . " bottles of beer on the wall.`n`n"
}

s .=    "No more bottles of beer on the wall,`n
        No more bottles of beer.`n
        Go to the store and buy some more,`n
        99 bottles of beer on the wall."

Gui, Add, Edit, w200 h200, %s%
Gui, Show, , 99 Bottles of Beer
Return
```

Listing 18: ‘99 Bottles of Beer’.

The slice is a modified version of a code from the ‘Rosetta Code’ website [10]. When running the script, a window of size 200 x 200 pixel appears, containing the full song.

Annotations: The `‘.’` operator, used frequently in this code, is for the concatenation of strings. An expression like `‘Var .= "xy"’` is a short way to write `‘Var := Var . "xy"’`.

3.5 Application Examples

In the following, application examples are described, giving an insight into what can be done with AutoHotkey.

Interaction with remote controls via WinLIRC

AutoHotkey is compatible with the open-source software WinLIRC [11], which is the Windows pendant to LIRC, the ‘Linux Infrared Remote Control Software’ [12]. WinLIRC can be used to transmit and receive infrared remote control signals to or from the computer, if an appropriate hardware is installed. That way, tasks on the computer can be controlled by a remote control. Similar, other devices can be controlled by the computer.

On the official website of AutoHotkey, a WinLIRC client script can be downloaded, which contains the setup to receive notifications from WinLIRC whenever a button on a remote control is pressed [13]. By changing the standard settings in this script, the user can individually automate programs by using an infrared remote control. Listing 19 shows a slice of the script, increasing the sound card’s volume by 5% when pressing the volume-up button on the remote.

```
VolUp:
SoundSet +5
return
```

Listing 19: A WinLIRC example.

Easy window control via NiftyWindows

NiftyWindows is a free tool designed by Enovatic-Solutions [14] and implemented with AutoHotkey. It gives control of basic window interactions like dragging, maximizing, minimizing, closing and many others.

One of the main features is the assignment of additional functions to the right mouse button. For example, windows are divided into a virtual 3x3-cell grid: the center cell is used to move the window around by clicking into the cell and dragging the window with the right mouse button. The same way, windows can be resized when using the right mouse button on the other eight cells.

Navigating the Start Menu

The ‘Seek’ script, created by the author Phi, can be used to traverse the Windows Start Menu [15]. By specifying a case-insensitive phrase or keyword, the script tries to find a match within the programs or directories of the Start Menu, providing a faster access for the user.

4 Considerations

In this section we discuss the Turing completeness of AutoHotkey and its integration in Windows. We compare it to other languages, and talk about its significance and alternatives.

4.1 Turing Completeness

AutoHotkey is Turing-complete. The language provides conditional branching (e.g. `‘if - else’` constructs), infinite looping (e.g. `‘while’` loop) and recursion. Furthermore, dynamic memory allocation is supported (for example when manipulating strings [6]), although there are no commands available to the user to explicitly ‘control’ memory allocation in AutoHotkey.

4.2 Integration in Windows

AutoHotkey is written in C++ and makes ample use of the Win32 API. This way, the user can write applications correlated to the Windows API, for example whenever creating a GUI in AutoHotkey.

4.3 Necessity of Creation

AutoHotkey was developed to provide an automation scripting-language for Windows systems, supporting a specific set of features, that at this time weren't covered by other languages or Windows itself. The utility was – and is still – downloaded by users, showing that these features were ‘needed’ in some way, also if this primarily means simplifying personal tasks.

4.4 Differences to similar Languages

When considering well-known programming languages, AutoHotkey can best be compared to the scripting language Microsoft Batch. It was designed to ease regular tasks by using scripts, that are executed by the command interpreter. The .bat files are of a plain text format and can be created with the help of any editor similar to AutoHotkey. Regarding syntax, there are some similarities (as for example in the use of `‘%’` signs to enclose variables), but the command set is different and also more difficult to learn.

Another comparison can be drawn from AutoHotkey to the general-purpose programming language BASIC. Both share the intention to provide an easy to learn language and have a similar, uncomplicated syntax, with minimalistic use of delimiters. BASIC however was developed to become widespread, and was not primarily intended for personal use as AutoHotkey.

AutoHotkey also includes other elements resembling aspects of well-known languages (among them different operators, as for example `‘:=’`, also known from Pascal, or `'!='`, used not only in C and Java). These however seem to be chosen arbitrarily and without deeper reasons.

Finally, it is natural to compare AutoHotkey to AutoIt v2, from which it was derived. Both share most basic commands, and AutoIt v2 scripts can be run by AutoHotkey with only few exceptions. Important differences are described in detail in the "Notes for AutoIt v2 Users" chapter on the AutoHotkey website [16] (for example, the 'break' command is used in AutoIt to enable or disable a user to exit a running script by the tray icon, whereas in AutoHotkey it is used to break out of loops). In comparison to AutoIt v2, AutoHotkey provides many additional commands due to new features, as for example hotkeys or hotstrings.

However, with 'v3' AutoIt now includes features that cover most functionality of AutoHotkey (among them hotkey support). The syntax is more structured as in AutoHotkey, and is clearly intended to fit for bigger Windows applications. Nevertheless it is not explicitly preferred over AutoHotkey, when considering user downloads. AutoHotkey is smaller and, especially for non-programmers, easier to use, also if its syntax is in some cases confusing.

4.5 Alternatives to AutoHotkey

AutoIt was already mentioned several times in this article, because it is kind of a predecessor to AutoHotkey. It is in general the more structured language and offers a broader set of possibilities, intended for larger applications.

Another alternative to AutoHotkey is the scripting language IronAHK, which is a complete rewrite of AutoHotkey, developed by members of the community and released since 2010 [17]. It also works on Linux or MAC and provides additional features to the original. However, it is only partially documented by now.

On Linux, the tool AutoKey was developed to cover some of the features that AutoHotkey provides for Windows [18]. It was however never intended as a full replacement for it and is also not a proprietary language (scripts are written in Python).

5 Conclusion

Chris Mallet's intention was to develop an easy to learn language that facilitates work on Windows systems. He also reached that goal, concluding from the fact that AutoHotkey is still downloaded by now. However, in contrast to AutoHotkey, AutoIt is still actively developed and already covers most features of AutoHotkey. It is therefore mainly a question of taste, when choosing AutoHotkey over AutoIt as automation language on Windows. In many cases however, the features provided by AutoHotkey may not be sufficient anymore.

References

[1] Chris Mallet. AutoHotkey. http://www.autohotkey.com/, 2004 - 2012 (accessed 5 February 2013).

[2] Chris Mallet. Changelog for AutoHotkey. http://www.autohotkey.com/changelog/, 2004 - 2012 (accessed 5 February 2013).

[3] Jonathan Bennett. AutoIt. http://www.autoitscript.com/, 1999 - 2012 (accessed 5 February 2013).

[4] Jonathan Bennett. AutoIt forum, thread "Licensing Opinions". http://www.autoitscript.com/forum/topic/7204-licensing-opinions/, 02 January 2005 (accessed 5 February 2013).

[5] Chris Mallet. AutoHotkey forum, thread "My status and website changes". http://www.autohotkey.com/board/topic/58864-my-status-and-website-changes/, 10 October 2010 (accessed 5 February 2013).

[6] Chris Mallet. AutoHotkey, chapter "Variables and Expressions". http://www.autohotkey.com/docs/Variables.htm, 2004 - 2012 (accessed 5 February 2013).

[7] Chris Mallet. AutoHotkey, chapter "Functions". http://www.autohotkey.com/docs/Functions.htm, 2004 - 2012 (accessed 5 February 2013).

[8] Chris Mallet. AutoHotkey, chapter "Send". http://www.autohotkey.com/docs/commands/Send.htm, 2004 - 2012 (accessed 5 February 2013).

[9] Chris Mallet. AutoHotkey, chapter "GUI". http://www.autohotkey.com/docs/commands/Gui.htm, 2004 - 2012 (accessed 5 February 2013).

[10] '99 Bottles of Beer'. http://rosettacode.org/wiki/99_Bottles_of_Beer, 2012 (accessed 5 February 2013).

[11] Jim Paris and Scott Baily. WinLIRC. http://winlirc.sourceforge.net/, 1999 - 2001 (accessed 5 February 2013).

[12] Christoph Bartelmus. LIRC - Linux Infrared Remote Control. http://www.lirc.org/, 1999 - 2002 (accessed 5 February 2013).

[13] Chris Mallet. AutoHotkey, chapter "WinLIRC Client". http://www.autohotkey.com/docs/scripts/WinLIRC.htm, 2004 - 2012 (accessed 5 February 2013).

[14] Enovatic-Solutions. NiftyWindows. http://www.enovatic.org/products/niftywindows/introduction/, (accessed 5 February 2013).

[15] Phi. Seek. http://www.autohotkey.com/docs/scripts/Seek_%28SearchTheStartMenu%29.htm, 2005 (accessed 5 February 2013).

[16] Chris Mallet. AutoHotkey, chapter “Notes for AutoIt v2 Users”. `http://www.autohotkey.com/docs/AutoIt2Users.htm`, 2004 - 2012 (accessed 5 February 2013).

[17] Tobias Kapp. IronAHK. `http://www.ironahk.net/`, 2010 (accessed 5 February 2013).

[18] AutoKey. `http://code.google.com/p/autokey/`, (accessed 5 February 2013).