Was kann AutoHokey?

AutoHotKey hat drei Grundfunktionen mit sehr ähnlicher Syntax:

Hotkeys

Einer Taste oder einer Tastenkombination eine bestimmte Funktion zuweisen (Programm aufrufen). Man kann mit einem Hotkey auch mehrere Programme ausführen.

Syntax
```
<Tasten>::Run <Programm>
```

Remapping

Tasten umkodieren. So, wie beim ersten Beispiel gezeigt, wird einer Taste oder einer Tastenkombination ein anderer Wert zugewiesen. Das funktioniert auch mit Maustasten oder Joystick-Tasten

Syntax
```
<Taste(n)>::Send <Tasten>
<Taste(n)>::<Taste>
```

Hotstrings

Gemeint ist die Substitution von Abkürzungen durch einen Volltext, zum Beispiel *„mfg."* durch *„mit freundlichen Grüßen"*

Syntax
```
::<Kürzel>::<Text>
```

Da aber ***AutoHotKey*** neben diesem Kürzelsystem auch über eine eigene Programmiersprache ***AHK*** mit Schnittstellen zum Dateisystem, zur Registry usw. verfügt, kann man es zum Programmieren verschiedenster Problemstellungen verwenden.

Die entstehenden Skripts können entweder als unabhängige Skripts oder auch als ein gemeinsames Skript in einer Datei ausgeführt werden.

Skripts können in eine Exe-Datei kompiliert werden und sind danach portabel und nicht mehr von der Installation von ***AutoHotKey*** auf einem Rechner abhängig.

Die Möglichkeiten sind derart vielfältig, dass man nur raten kann, in der deutschspracheigen Hilfe-Datei zu blättern und sich von den zahlreichen Beispielen, auch im Forum anregen zu lassen.

Wie arbeitet AutoHotKey?

Wie das Programm eine Tastatur beeinflusst, bestimmen kleine Konfigurationsdateien (***AHK***-Skripts), die als Parameter an das Programm übergeben werden.

Jeder Hotkey, jedes Mapping und jeder Hotstring kann in einer eigenen Datei mit der Endung **.ahk** gespeichert werden oder man kann auch alle oder einige in einer einzigen Datei zusammenfassen; einfach innerhalb einer AHK-Datei aneinanderreihen.

Es können beliebig viele solcher Skripte geladen und wieder entfernt werden ohne den PC neu starten zu müssen

Die so definierten Tastenkürzel oder auch Textbausteine funktionieren in allen Programmen.

Man kann beliebig viele solcher Skripte laden. Jedes Skript äußert sich in einem Symbol im Infobereich rechts in der Taskleiste. Man kann jedes Skript temporär ausschalten oder entladen. Das Testen der Skripts in daher überaus einfach.

Arbeitstechnik

Um ein Skript (Textdatei mit Endung **.ahk**) zu laden, klickt man mit der rechten Maustaste darauf. Im Kontextmenü finden sich gleich am Anfang drei Einträge, die AHK-Dateien betreffen:

- *Run Script*
- *Compile Script*
- *Edit Script*

Run Script lädt das Script und fügt ein Symbol in die Taskleiste ein. *Compile Script* erzeugt eine Exe-Datei und Edit Script öffnet das Skript im Editor.

In der Taskleiste werden nun so viele AHK-Symbole angezeigt als Skripts geladen worden sind. Man kann sie durch Überfahren mit der Maus unterscheiden, weil sie dann in einer Sprechblase den Namen bekanntgeben. Über einen Klick mit der rechten Maustaste sieht man die Menüpunkte *Open, Help, Window Spy, Reload, Edit, Suspend, Pause und Exit*. *Help* öffnet eine ***AHK***-Hilfedatei, *Window Spy* gibt einige Daten zum aktuellen Fenster und zur Mausposition bekannt, mit *Reload* kann das Skript neu geladen werden (etwa, wenn man es im Editor korrigiert hat), mit *Edit* öffnet man das Skript im Editor, mit *Suspend* wird das Skript temporär ausgeschaltet, mit *Pause* wird das aktuelle Skript angehalten und mit *Exit* wird das Skript entladen.

Das Testen eines Skript verläuft also so, dass man den Text des Skripts im Editor verändert und dann in der Statusleiste beim Symbol für dieses Skript den Menüpunkt *Reload* aktiviert. Keine Registry-Einträge, kein Reboot des Rechners erforderlich.

Wenn man intensiv testet, ist sogar der Griff zur Tastleiste lästig. Man kann dann an den Anfang des Skripts einige Kodezeilen einfügen und das Skript neu starten.

```
SetTimer,UPDATEDSCRIPT,1000
UPDATEDSCRIPT:
FileGetAttrib,attribs,%A_ScriptFullPath%
IfInString,attribs,A
{
FileSetAttrib,-A,%A_ScriptFullPath%
SplashTextOn,,,Script aktualisiert,
Sleep,500
Reload
}
Return
```

Diese Kodezeilen sorgen dafür, dass das Skript bei jedem Speichern des Textes automatisch neu geladen wird.

Die Sprache *AHK*

Zunächst muss man wissen, dass in ***AHK***, der Sprache von ***AutoHotKey*** jede Taste am PC und auch jede Aktion der Maus durch ein reserviertes Wort repräsentiert wird. **Space** ist die Leertaste, **PgUp** ist die BildNachOben-Taste, usw. Die vollständige Liste: findet sich auf der vorigen Seite.

Alphazeichen, Zahlen und Zeichen brauchen meist keinen eigenen Namen. Zum Beispiel: **b** ist die B-Taste und **5** die 5-Taste.

Jetzt muss man noch wissen, dass die Umschaltetasten in ***AutoHotKey*** der Einfachheit halber abgekürzt werden mit

Taste	***AHK***-Kürzel
⇧	+
Strg	^
Alt	!
AltGr	<^>!
⊞	#

Weitere Sprachelemente werden in den Beispielen vorgestellt. Eine Gesamtübersicht über die Sprache findet sich in der Hilfedatei zu ***AutoHotKey***.

Bevor wir daran gehen, Hotkeys und Remapping zu implementieren, müssen wir uns eine Übersicht verschaffen, welche Möglichkeiten dazu überhaupt bestehen. Es muss ja sichergestellt werden, dass ein Hotkey oder auch ein Remapping möglichst nicht mit einem bereits in einem Programm definierten Hotkey konkurriert.

Wie viele verschiedene Kodes kann man einer Taste zuordnen?

Verwenden wir als Beispiel die Taste Q. Diese Taste kann mit jeweils einer von fünf Umschaltetasten sechs Zeichen generieren.

Taste	Zeichen
Q	q
⇧ Q	Q
Strg Q	Strg-q
Alt Q	<unbelegt>
AltGr Q	@
⊞ Q	Suchmenü

Zwar sendet eine Tastatur in allen diesen Fällen für die Taste Q selbst nur einen Kode aber durch das gemeinsame Drücken mit den jeweiligen Umschaltetasten, die auch einen Kode senden, ergibt sich immer ein anderes resultierendes Zeichen.

Aber wie viele Zeichen könnte die Taste Q insgesamt erzeugen?

Da es fünf Umschaltetasten gibt: ⇧ Strg Alt AltGr und ⊞ und dazu auch die grundlegende Möglichkeit keine Umschaltetaste zu benutzen, kann man mit einer Taste 2^6=64 Zeichen generieren.

Es gibt etwa 64 Zeichen-Tasten. Daher kann eine Tastatur etwa 4096 Zeichen generieren. Theoretisch, denn wer wird schon gerne das Zeichen über die Tastatur erzeugen wollen, das sich aus der Tastenkombination ⇧ Strg Alt AltGr ⊞ Q ergibt.

Wie würde man also einer Taste über Hotkey diese verschiedenen Möglichkeiten verpassen? Welche Tastenkombination soll man wählen?

Die meisten Shortcuts nutzen aus praktischen Gründen auch nur ein Umschaltzeichen, manchmal auch zwei, selten drei aber nicht mehr Kombinationen aus.

Man ist ja nicht allein am PC, da gibt es auch das Betriebssystem, das uns viele Tastenkürzel vorgibt und jedes aktive Programm beansprucht auch noch einen Satz von Tastenkürzel.

Wenn also das eigene Tastenkürzel systemweit Gültigkeit haben soll, darf es nicht mit den bereits definierten Kürzel konkurrieren.

Das Betriebssystem konzentriert sich auf Tastenkombinationen mit der Windows-Taste. **(siehe umseitige Tabelle)**

Mehrere Programme mit einem Hotkey ausführen

```
RAlt & c::
{
   Run, c:\Programme\Mozilla Firefox\firefox.exe
   Run, c:\Programme\Mozilla Sunbird\sunbird.exe
   Run, c:\Programme\Mozilla Thunderbird\thunderbird.exe
}
return
```