FINDCLICK DOCUMENTATION

Function by Berban
Last updated May 6, 2017

FindClick is a standard-library compatible image searching utility compatible with AutoHotkey Basic or _L.

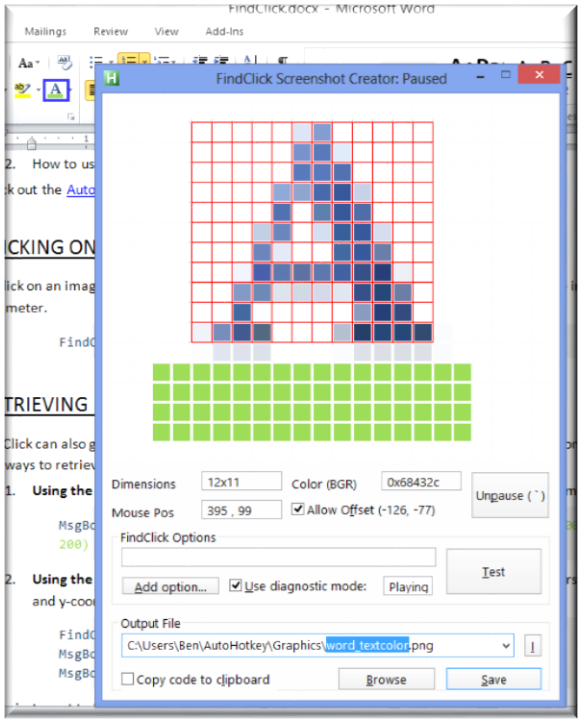
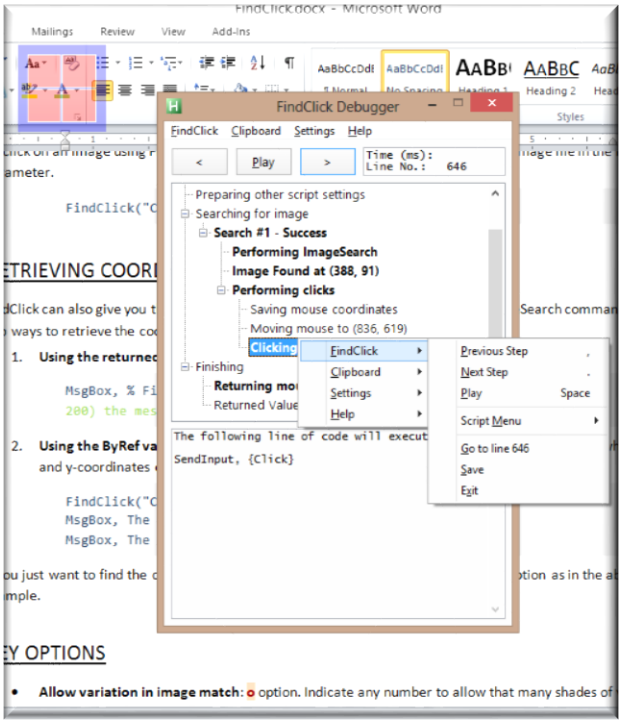


Image Creator mode



Diagnostic mode

This documentation is a work in progress. Some sections may be empty and will be expanded later.

TABLE OF CONTENTS

FindClick Documentation *1*

Table of Contents *2*

Quick-Start Guide *4*

AutoHotkey function basics 4
Clicking on an image 4
Retrieving coordinates of an image 4
Key options 4
Creating an image 5

ImageSearch Mode *5*

ImageFile Parameter **5**
Search using a pixel color 5
Format of pixel color 6
Comparison with PixelSearch 6
Search using a preexisting image file 6
Omit directory using %DefaultDirs% 6
Omit extension using %DefaultExts% 7

Options Parameter **7**
Using the options parameter 7
Modifying Default/User Default values 8
User Configurations for options 8
List of options 8
o – ImageSearch Options 9
a – Search Area Modifications 9
r – Relative to Window Coords 9
x – X Offset 9
y – Y Offset 9
n – Number of clicks (No click) 9
e – Find Every Image 10
w – Wait until image is found 10
dx – Diagnostic Mode 10
k – Keystroke(s) 10
Stay – Do not restore mouse 10
Count – Return found count 11
d – Direction Of Search 11
m – SendMode 11
Func – Function Callout 11
Sleep – Sleep between clicks 11
t – Image Tracking 11

Silent – No Dialogs 12
Center – Start at center of image 12
Delim – Delimiter for multiple images 12
f – Format of output string 12
CharX – Character spaceholder for x-coordinate 12
CharY – Character spaceholder for y-coordinate 12
CharN – Character spaceholder for image instance number 12

Image Creator Mode 13

Starting the image creator 13
Searching for a non-existent image file. 13
Explicitly calling the image creator 13
Using the image creator 13
Magnify the area you want 13
Selecting a region from the magnifier 14
Allow offset 14
Testing 14
Saving your image 14
Copy code to clipboard 14
Modifying image creator settings 14

Diagnostic Mode 15

Initiating diagnostic mode 15
Using the dx option 15
After an error 15
Using the debugger gui 15

ImageSearch Suggestions 16

Choosing an image 16
Performance 16

Examples 17

Click on Text in Google Chrome 17
Duplicate Tab in Google Chrome 17
Seek in Spotify 18

Frequently Asked Questions 19

Buttons that look different when the mouse hovers 19
Choosing among multiple results 19

Special Thanks 20

QUICK-START GUIDE

AUTOHOTKEY FUNCTION BASICS

Users with some experience should already be familiar with this part, but if these steps sound strange to you then be sure to check them out.

1. Including a function in your script
2. How to use functions

Check out the AutoHotkey functions documentation for more information.

CLICKING ON AN IMAGE

To click on an image using FindClick, simply call the function and put the path to your image file in the first parameter.

```
FindClick("C:\Images\My Image.png")
```

RETRIEVING COORDINATES OF AN IMAGE

FindClick can also give you the coordinates of the onscreen image much like the ImageSearch command. There are two ways to retrieve the coordinates.

1. **Using the returned value**: The function will return the coordinates delimited by a comma.

```
MsgBox, % FindClick("C:\MyImage.png") ; If the image is found at (100,
200) the message box should display "100,200"
```

2. **Using the ByRef variables**: The 3rd and 4th parameters of FindClick are ByRef parameters in which the x- and y-coordinates of the image location will be stored if the search is successful.

```
FindClick("C:\MyImage.png", "n", xCoord, yCoord)
MsgBox, The x-coordinate of the image is %xCoord%
MsgBox, The y-coordinate of the image is %yCoord%
```

If you just want to find the coordinates of the image without clicking on it, use the **n** option as in the above example.

KEY OPTIONS

- **Allow variation in image match**: **o** option. Indicate any number to allow that many shades of variation.

```
FindClick("C:\MyImage.png", "o12") ; Allows 12 shades of variation
```

- **Find all instances of an image**: `e` option. All coordinate pairs will be returned by the function, delimited by newline (`n) characters.

```
MsgBox, % FindClick("C:\MyImage.png", "e") ; Will display each location
at which an image is found on its own line
```

- **Relative to active window only**: `r` option.

```
FindClick("Image", "r") ; Will only search for image inside the active
window's borders
```

- **Click several pixels away from an image**: `x` and `y` options.

```
FindClick("Image", "x10 y-20") ; Will click 10 pixels to the right of
and 20 pixels above the center of the image, if found
```

See the section on Options for more info.

CREATING AN IMAGE

To use FindClick's built-in image creator, which is optimized for the sort of small and precise images that ImageSearch uses, call the function with no parameters.

```
FindClick() ; Will display image creator window
```

More info in the relevant section below.

IMAGESEARCH MODE

ImageSearch mode is the primary intended use for this script.

IMAGEFILE PARAMETER

The first parameter in FindClick is the graphical element that you are searching for. This can either be a preexisting image file or a colored region.

SEARCH USING A PIXEL COLOR

FindClick can emulate the behavior of PixelSearch if you specify a pixel color for the ImageFile parameter. This can be useful if the color you are searching for might change while the script is executing. It can also help reduce unnecessary image file clutter.

Format of pixel color

1. Asterisk (*) — This filepath-incompatible character tells FindClick that you mean a color and not a file.
2. Pixel Color — Pixel color must be a hexadecimal value and not a color name. Whether it is interpreted as a RGB or BGR value depends on the value assigned to %UseRGB% at the top of the code. The 0x prefix is optional.
3. Dimensions (optional) — Following the color code may optionally come a non-numeric character and then a Width x Height value. This tells FindClick to find a region of this width and height of the given color as opposed to a single pixel.

```
; Finds a white pixel and clicks on it
FindClick("*0xFFFFFF")
; Finds a uninterrupted region of white pixels that is at least 10
pixels wide and 20 pixels high
FindClick("*FFFFFF 10x20")
```

Comparison with PixelSearch

Despite its similarity to PixelSearch, searching for a pixel color with FindClick still uses AutoHotkey's ImageSearch technique. This introduces some key differences between FindClick and PixelSearch:

- Compatible with Windows Aero and Windows 8. Starting in Windows Vista, Microsoft introduced a new way to render the desktop called desktop composition. This breaks PixelSearch; however, ImageSearch (and by extension FindClick) is unaffected. In Vista and 7 you can disable destop composition but it cannot be turned off in Windows 8. FindClick can be used as a workaround to this issue.
- Regions may be used instead of pixels. You can search for an area of color instead of a single pixel, which helps avoid false positives in photos or gradients.
- An image file must still be created on disk. FindClick uses GDI+ to create an image of the specified color and size before performing the search. The image file is stored in the windows temporary directory. This is a relatively CPU-intensive step (<10 milliseconds) but only need be done once per image file until temporary files are cleared.
- Less finicky with transparency. Some of PixelSearch's shortcomings are avoided by using ImageSearch (see the "Remarks" section in the PixelGetColor documentation for more info.)

SEARCH USING A PREEXISTING IMAGE FILE

The more common use of FindClick is to specify an image file on your hard disk as the item to search for. This is the same as AutoHotkey's ImageSearch command (around which FindClick is based.)

One difference between the ImageFile parameter in FindClick and ImageSearch is that FindClick allows you to omit parts of the file name which the function will fill in.

Omit directory using %DefaultDirs%

At the top of the code for FindClick a variable %DefaultDirs% is declared. %DefaultDirs% may contain a pipe-delimited list of file directories. FindClick will treat each of these directories as if it were an additional working directory. If a relative path is given for an image file, and it is not found in the working directory, the function will check for that file in each of the %DefaultDirs% in the order in which they are written.

```
; Any image in C:\Images or on the desktop need not have a full path
; This includes subfolders of these locations: for instance for
"C:\Images\Folder\File.png" you may indicate "Folder\File.png"
DefaultDirs = C:\Images|%A_Desktop%
```

Simply edit the declaration of %DefaultDirs% to include your most commonly used folders.

One directory that %DefaultDirs% is prepopulated with is `%A_AhkPath%\..\Graphics`. This represents a subfolder of your AutoHotkey folder named "Graphics." I find this is a convenient place to keep all my FindClick-related graphics, allowing them to be shared among all uncompiled scripts.

Omit extension using %DefaultExts%

Likewise, the file extension can be omitted if it is among the pipe-delimited extensions given in %DefaultExts%. This is mostly for convenience as most images used for FindClick will be of the .png format, so it's unnecessary to specify this each use.

```
FindClick("C:\Image") ; If "C:\Image" does not exist, the function will
check for "C:\Image.png" and "C:\Image.bmp" and use these if present.
```

OPTIONS PARAMETER

USING THE OPTIONS PARAMETER

The Options parameter in FindClick is similar to the Options parameter found in the AutoHotkey Gui command (for instance, when using Gui, Show).

The valid options for FindClick can be found about 25 lines from the top of the script under the heading "Default Options." (They are also listed below.) Each option will be assigned a certain value when you call the function.

- **Default value** The option assumes this value if it is omitted entirely from the options string. An option's default value can be found in the first column of declarations under "Default Options" in the code.

```
FindClick("Image", "x20 e") ; All options except x and e assume their
default values.
```

- **User default** This value is assumed if the user includes the option name in the option parameter but does not follow it with a new value string. The user default is basically intended to be used as a sort of "favorite value" for that particular option. The user defaults can be found in the second column of declarations under "Default Options."

```
FindClick("Image", "x20 e") ; e assumes the user default value.
```

- **Other/Custom** To pass any value besides those found in the two columns at "Default Options" in the code, simply immediately follow the option with the desired value.

```
FindClick("Image", "x20 e") ; x takes "20" for its new value.
```

If the string you are trying to give contains spaces, surround the entire string with double-quotes. Inside the quoted string, two double-quotes resolve to a single double-quote. (This doubling is not necessary not necessary unless the whole string is quoted.)

```
Options = x20 r"string has ""spaces"" and ""quotes""" e
FindClick("Image", Options) ; r will receive the following value:
; string has "spaces" and "quotes"
; Traditional declaration was used here because in expression mode
double-quotes would need to be doubled yet again, which can be
confusing.
```

If you are unsure that FindClick understood your options string correctly, you can always use **dx** (diagnostic mode) which will display the interpreted values for all script options.

Modifying Default/User Default values

Any of the "Default" or "User Default" declarations can be easily modified inside the FindClick code under the `Default Options` header. (To find this, search the code for the string "Default Options".)

For instance, the line containing the defaults for the **m** (SendMode) option normally appears below:

```
, m := "Input",          m_user := "ControlClick"      ; SendMode
```

However if you would like FindClick to use the SendPlay mode by default unless you specify otherwise, you can change that line to the following:

```
, m := "Play",           m_user := "ControlClick"      ; SendMode
```

User Configurations for options

If a certain combination of options is used frequently you can store it in the FindClick code as a sort of "favorite" option string. To do this, scroll to the "Default Options" section of code as described above. At the bottom you will see an example user configuration like this:

```
UserConfig_foo = r"Test Window" mControlClick
```

When invoked, this configuration will simply add the given text to the options string: in this case, **r** will be set to `Test Window` and **m** will be set to `ControlClick`.

You can edit a given user configuration simply by changing the text stored in the variable. You can also change the name by changing the part of the variable after the underscore.

```
UserConfig_a = o20 x15 ; Now the user configuration !a will set 0 to 20
and x to 15
```

To invoke a user configuration, add it to the options string preceded by an exclamation mark (`!`).

```
FindClick("Image", "e !a") ; Invokes user configuration !a
```

LIST OF OPTIONS

Sorted loosely in order of how useful they are.

o – ImageSearch Options

What to give comma-delimited string of imagesearch options

Description The optional parameters for ImageSearch, as shown in the AutoHotkey documentation. Use a comma to separate options instead of a space. You may omit the asterisk (`*`). For example: `oTransBlack,20` makes black transparent and allows 20 shades of variation.

a – Search Area Modifications

What to give `x[,y,w,h]` or `mn`

Description Region within which to search for the image. Indicate either of the following:
1) A comma-delimited list in the format `x,y,w,h`. List items may be blank or absent to leave that value unchanged. Coordinates are relative to the window if **`r`** is used. For w and h, use `+` and `-` to indicate a change to the normal endpoint (depending on a, either the edge of the screen or active window). For instance, `a,,-300` will search the entire screen (or window if **`r`** is used) except for the last 300 pixels of the right side – the width of the search area has been reduced by 300.
2) The letter `m` and then a number to search a square region centered at the cursor position and with a width and height twice that number.

r – Relative to Window Coords

What to give window title criteria OR a window hwnd

Description The search area becomes the area of the given window instead of the entire screen area. This is useful when scanning for an element which only appears on a particular window – reducing the search area improves performance. Use of the **`a`** option in conjunction with **`r`** will apply these offsets to the window coordinates.

x – X Offset

What to give any integer

Description This many pixels will be incremented to the click coordinates before clicking. A positive **`x`** value will click left of the image, a negative value will click to the right. This change will also show up in the string that is returned by the function and the coordinates stored in the ByRef variables.

y – Y Offset

What to give any integer

Description This many pixels will be incremented to the click coordinates before clicking. A positive **`y`** value will click below the image, and a negative value will click above it. This change will also show up in the string that is returned by the function and the coordinates stored in the ByRef variables.

n – Number of clicks (No click)

What to give any integer

Description The number of times to click on each image. Indicate `0` (the user default) to do no clicking whatsoever and just return the coordinates of the found image(s). In this sense, specifying just `n` is like saying “no click”. Without clicks, the **`m`**, **`Sleep`**, and **`k`** options are irrelevant, however, the **`x`** and **`y`** values will still be added to the output coordinates.

e – Find Every Image

What to give positive fraction between 0 and 1

Description If this option is nonblank then FindClick will find (and click) EVERY instance found. This means that after any successful ImageSearch execution the script will queue up a new call of ImageSearch which represents the screen area left unsearched by the first call. (Note that this therefore makes FindClick slower for any use in which you don't expect to find more than one image.)
The value of **e** signifies the overlap between search areas, in terms of the fraction of the image width and height. Overlap should normally be around `0.9` except if the image is uniform throughout, e.g. a 10x10 square image containing mostly white pixels. The **t**, **d**, and **Count** options all require the "find every image" behavior as part of their execution and so using any of these other options implies **e**, however, you may still specify a custom value for **e** to change the overlap for these other behaviors.

w – Wait until image is found

What to give number of milliseconds [, number of milliseconds]

Description If the image is not found, the function will wait this many milliseconds for it to appear before returning. You may add a comma and then a second number which indicates the delay in milliseconds between subsequent ImageSearches. For example `2000,50` means the function will look for the image every 50ms until either the image is found or 2000ms elapse, for a maximum of 40 ImageSearch operations. If omitted, this delay will be the smaller of either the wait time divided by 10 or 100ms.

dx – Diagnostic Mode

What to give comma delimited list of diagnostic mode options

Description Will produce an AutoHotkey GUI before the function completes showing a step-by-step history of how the function executed. The value of the string passed to the **dx** option may be used to send certain flags to the debugger GUI before it displays. For more info on this see the section on the FindClick debugger.

k – Keystroke(s)

What to give key to send (in the same format as the AutoHotkey Send command)

Description Indicate the keys to press (if any) when each image is found. **k** can include multiple keypresses and even non-mouse keys, for instance, `^{Space}`. If **m** (SendMode) is ControlClick then the format is different (see the ControlClick documentation) and a left click will be assumed if an incorrect **k** is given.

Stay – Do not restore mouse

What to give true or false (`1` or `0`)

Description Normally after any clicks executed by FindClick the mouse will be immediately returned to its initial location. If **Stay** is true then the mouse will NOT return to its original position after FindClick finishes.

Count – Return found count

What to give: true or false (`1` or `0`)

Description: If **`Count`** is true then the function will return the number of items found instead of their coordinates. **`e`** is assumed if absent.

d – Direction Of Search

What to give: `left`, `right`, `bottom`, or the first letter of any of these words

Description: This setting attempts to emulate the mode of PixelSearch where you can search from right to left or top to bottom – for instance, specifying `Bottom` will find the image closest to the bottom of the search area. However, to accomplish this the script needs to first find every image and then choose the coordinate pair that is furthest in the requested direction, making it a time-consuming process.

m – SendMode

What to give: `event`, `play`, `input`, `default`, `controlclick`, or the first letter of any of these words

Description: The send mode (or first letter thereof) to use for clicks, i.e. `Input`, `Play`, or `Event`. Specify `ControlClick` (or `c`) to use a controlclick instead of a simulated keystroke. If **`m`** is blank or the letter `d` (for default), the current SendMode will be used.

Func – Function Callout

What to give: the name of a function in the script

Description: Each time an image is found, instead of clicking on the image or executing keystrokes the given function will be called. The function must accept at least two parameters: the x-coordinate of the image will be passed in the first parameter, and the y-coordinate will be passed in the second parameter. Note that any thread settings such as CoordMode that are changed within the function will carry over to FindClick when the function is done.

Sleep – Sleep between clicks

What to give: any number of milliseconds

Description: Number of milliseconds to sleep between each click if **`n`** > 1 or **`e`** and multiple images found.

t – Image Tracking

What to give: percent OR number of pixels

Description: This option is used to improve performance if an image will be found nearby where its last location. The script will first search nearby the last found position and if it is not found there then the rest of the area will be searched. Indicate a percentage to search within that percentage of the search area in each direction. For instance, the string `20%` will first search a box that begins 20% of the distance between the starting x position and the last found x position, and likewise for the other three directions. Indicating a number will simply search a box that many pixels from the last found area before searching the rest of the s area. Indicate `0` to search EXACTLY the last found location first. If **`t`** is negative (including `-0`) then it will ONLY search in the region requested, and will not go on to search the rest of the screen if the image is not found there.

Silent – No Dialogs

What to give: true or false (`1` or `0`)

Description: Instead of displaying an error dialog when an error prevents the function from executing properly (for instance, if the image file is not found), it will silently return a blank string and set the ErrorLevel to the error message. The errors concerned should not appear during normal execution and so **`Silent`** is generally not recommended.

Center – Start at center of image

What to give: true or false (`1` or `0`)

Description: If **`Center`** is true then clicks will occur at the center of the image, i.e. its found position plus half its width and half its height. The **`x`** and **`y`** options treat this as 0, 0 – for instance, by default an **`x`** of `2` will click 2 pixels to the right of the image center. Indicate false to instead start at the top left corner of the image.

Delim – Delimiter for multiple images

What to give: any character or characters

Description: If **`e`** is used and multiple images are found then each image's info is separated by this character(s) in the returned string. The format of the image info is determined by the value of **`f`**.

f – Format of output string

What to give: template string that includes CharX and/or CharY and/or CharN characters

Description: The given string represents a template into which image x-coordinate, y-coordinate, and instance number will be substituted. Each time an image is found, the **`CharX`**, **`CharY`**, and **`CharN`** characters will be replaced with these values. The final string is returned by the function. See the examples section of the documentation for an example.

CharX – Character spaceholder for x-coordinate

What to give: any character

Description: Any instance of this character, if present, in the string given for **`f`** will be replaced with the image's x-coordinate.

CharY – Character spaceholder for y-coordinate

What to give: any character

Description: Any instance of this character, if present, in the string given for **`f`** will be replaced with the image's y-coordinate.

CharN – Character spaceholder for image instance number

What to give: any character

Description: Any instance of this character, if present, in the string given for **`f`** will be replaced with the order in which the image was found among all images found. This character is only relevant when **`e`** is being used and is therefore not normally included in **`f`**.

IMAGE CREATOR MODE

Built into the FindClick code is a single-purpose graphical interface for creating the sort of small image files that you will need for FindClick or ImageSearch. Using this tool is considerably faster and easier than using printscreen and mspaint.

STARTING THE IMAGE CREATOR

There are two main ways to bring up the FindClick screenshot creator GUI window:

Searching for a non-existent image file.

If you call FindClick with an image file that does not exist it will give an error dialog asking if you want to create the image. Simply select "Yes" in this dialog to go to the image creator. The output file path and image search options fields will be prepopulated with the values used in the original function call.

Explicitly calling the image creator

If FindClick is called with a blank image file then the GUI will appear with default settings.

```
FindClick() ; Summon the image creator GUI
```

To prepopulate an output file path, precede it with a caret `>` in the ImageFile parameter.

```
FindClick(">Suggested Name.png") ; If the path is relative, as it is
here, the user can choose which of the %DefaultDirs% to put the file in
```

More info on explicitly calling the image creator:

- Using two carets `>>` instead of one will switch the output file path groupbox control to a dropdown that the user can optionally edit. Using three carets `>>>` will disable editing.
- Prepopulate the options field by including text in the Options parameter.
- If the GUI is submitted and an image is created, FindClick will return True. If used, FoundX will contain the ultimate output file path and FoundY will contain the options the user entered.

USING THE IMAGE CREATOR

When the image creator appears you will see an array of pixels that magnify a region of the screen.

Magnify the area you want

As you move your mouse the display should update and show the area under your cursor. If this does not happen then the image creator is probably paused. Press the "Unpause" button and drag the cursor over the on-screen element you are interested in.

When the element of interest has appeared on the magnifier, press the pause hotkey to freeze the display. The default pause hotkey is backtick (`) but you can change this in the image creator settings section.

Selecting a region from the magnifier

The size of the magnified region is not large but your final image will likely be even smaller. To select a region for your image, click on one of the magnified pixels in the display. Move the mouse from the top-left corner of the desired region to the bottom-right corner and click again. The pixels in your region should now be surrounded by a pink box.

If you omit this step then the entire region will be used.

Allow offset

Sometimes the appearance of a graphical element will change when you hover the mouse over it. This means that hovering the magnifier over the image won't give the desired result.

To work around this, pause the magnifier and check the "Allow Offset" checkbox. This means that the blue magnified region will move when the mouse moves, as opposed to moving to the location of the mouse. If you try it out you'll get the hang of it.

Testing

You can test the image you've selected with FindClick by pressing the "Test" button. The image creator gui will hide and the script will search for the image on your screen. The contents of the "Options" edit field will be used as the FindClick options.

If the "Diagnostic Mode" checkbox is checked then the dx option will be appended to your options string when you perform the test. This means that the debugger gui will display once the test finishes. Simply close the debugger to return to the image creator.

Saving your image

When you press the "Save" button your selected image will be saved to the path in the "Output File" field.

The dropdown items in the "Output File" field are populated with the contents of %DefaultDirs% (described earlier). These are assumed to be the most likely places you'll be saving a file, however you may also press "Browse" to select another location.

Copy code to clipboard

If the "Copy code to Clipboard" checkbox is checked when you save the image file, AutoHotkey code for the given FindClick call will be stored in the clipboard. The ImageFile parameter will contain the path to the image file you just created, and the options parameter will contain the contents of the test options field.

MODIFYING IMAGE CREATOR SETTINGS

If you search the code of FindClick for the phrase "Screenshot Builder Settings" you can see a series of variable declarations that influence various aspects of the image creator. These settings include:

- Changing the text displayed on buttons (e.g. for changing the language)
- Changing accelerator keys (alt shortcuts)
- Changing the size of the magnifier (i.e. the number of pixels displayed)
- Changing the size & font of other elements
- Changing the default state of any of the checkboxes.

…among other settings. Simply find this part of the code to browse the settings that can be manipulated. All the settings are described in the function comments.

Note that I recommend changing AllowOffset to true once you have the hang of the function, as it is often necessary and starting out with it enabled saves time.

DIAGNOSTIC MODE

The diagnostic mode in FindClick is to be used whenever FindClick is not doing what you want it to do.

When the diagnostic mode is in effect, FindClick will execute normally and then display a GUI once the function has finished. This debugger window will attempt to walk the user through how the code executed. After all, the function is several hundred lines long, and it is easy to lose track of what is happening inside it.

INITIATING DIAGNOSTIC MODE

Using the dx option

The main way to tell FindClick to display the debugger is to use the `dx` option.

After an error

If FindClick encounters an error (e.g. if it cannot find the image file you specified), an error dialog will be displayed (unless the `Silent` option has been specified.) If you press "Debug" in that error dialog, the function will run again and display the debugger GUI when it encounters the error.

USING THE DEBUGGER GUI

The main feature of the debugger GUI is a treeview containing each step of the code.

- Bold steps have a graphical component. These include a preview of the image used, highlighting the search area, or demonstrating where clicks were performed.
- The box in the top left shows the amount of time spent on the selected step (and all substeps), allowing you to narrow down what is causing performance issues.
- Right-click or use the menus to take actions based on the selected step in the code, including copying the contents of relevant variables or opening the script to that location in the code.

(…this section is a work in progress…)

IMAGESEARCH SUGGESTIONS

AutoHotkey's ImageSearch is an incredibly useful tool; however, like many tools, there are several right and wrong ways to use it. The following are some possibly useful tidbits I've learned while using ImageSearch/FindClick for various tasks.

CHOOSING AN IMAGE

- **Your image should be unique**. Optimally, across whatever region of the screen that you are searching, there should only be one occurrence of the image (unless you are looking for multiple.) If there is more than one occurrence then you can try reducing the area to be searched. For instance if there is a graphic somewhere in the top of a window and another near the bottom, you can search with the options `r a,-200`. This means only the bottom 200 pixels of the window's height will be considered.
- The type of on-screen element is hugely important.
    - **Graphics are best**. By "graphic" I mean an element of an application that is stored as an array of colored pixels, i.e. like an image file. On a website, any element with an image src is a graphic. In other applications it's harder. Any icons, such as a printer or a pair of scissors, should be graphics.
    - **Gradients & colors are usually workable**. For instance, selected text can often be found by searching for the color of the selection highlight. A colored button might not be based on a graphic but if it's a fairly unique hue then you can use a snippet of its non-text area. If you are searching for a gradient then make sure **o** is more than 0 to allow for some variation.
      Note that gradients & colors will likely be your only option in a 3D game as everything will be rendered.
    - **Text is usually bad**. By "text" I mean data that is stored in Unicode or ASCII format. This is bad because each time the text is displayed the application might render it differently.
- **The image doesn't have to be what you're searching for**. Sometimes the thing you want to click on won't be unique. This is fine as long as there is a unique element nearby it that can anchor its position. For instance if you are trying to click on a search box, the white edit field background probably won't be unique, but the search button might be. Then you can use the **x** and **y** options to slide over to the place you actually want to click.
- **Smaller is better**. I barely ever use image files larger than 10x10. Usually they are smaller than that. Your image should only contain as many pixels as you need to guarantee it'll be unique on the screen. Too many beyond that just increases the likelihood that your image won't match.

PERFORMANCE

- The amount of time an ImageSearch command takes is roughly proportionate to the area of the search (assuming an image is not found) and the shades of variation allowed.

EXAMPLES

CLICK ON TEXT IN GOOGLE CHROME

AutoHotkey can't be used with COM to automate Google Chrome like with Internet Explorer. However, if you want to click on a textual element (a very common task with automating web tasks) you can take advantage of the fact that using Ctrl+F in Chrome will always highlight the text with the same color. This method will even scroll to the text if it isn't currently visible. However, unlike with COM, Chrome must be the active window.

```
ChromeClick(Text)
{
    Send ^f^f ; open the find dialog box – send twice to be safe
    SendInput {Raw}%Text% ; type in the search term
    ; Send {Enter %n%} ; Normally this will only find the first
    occurrence of the desired word. If you know you need the 2nd or 3rd
    you could add a line like this.
    If !FindClick("*0x3296FF 3x2", "r a,73 k{Esc}{Click} y5 w500,0") {
           Send {Esc} ; if the text isn't found, close the find box
           SoundPlay, %A_Windir%\Media\Windows Ding.wav ; ding for error
    }
}

/* Explanation of FindClick parameters:
*0x3296FF 3x2 = the color of the highlight around the found text item
k{Esc}{Click} = instead of clicking, the function will press escape (to
    close the find box) and then click. If you click first the find box
    will no longer have focus and will not close when escape is pressed.
y5 = click a tiny bit below the image because it will usually find the
    top edge of the highlight
w500,0 = wait 500ms in case chrome doesn't immediately find the word. 0
    for maximum performance during this short period.
*/

#IfWinActive Google Search ahk_class Chrome_WidgetWin_1 ; example hotkey

^n::ChromeClick("Next") ; go to next page in Google search results
```

DUPLICATE TAB IN GOOGLE CHROME

Mimics how in Internet Explorer you can press Ctrl+K to duplicate the current tab.

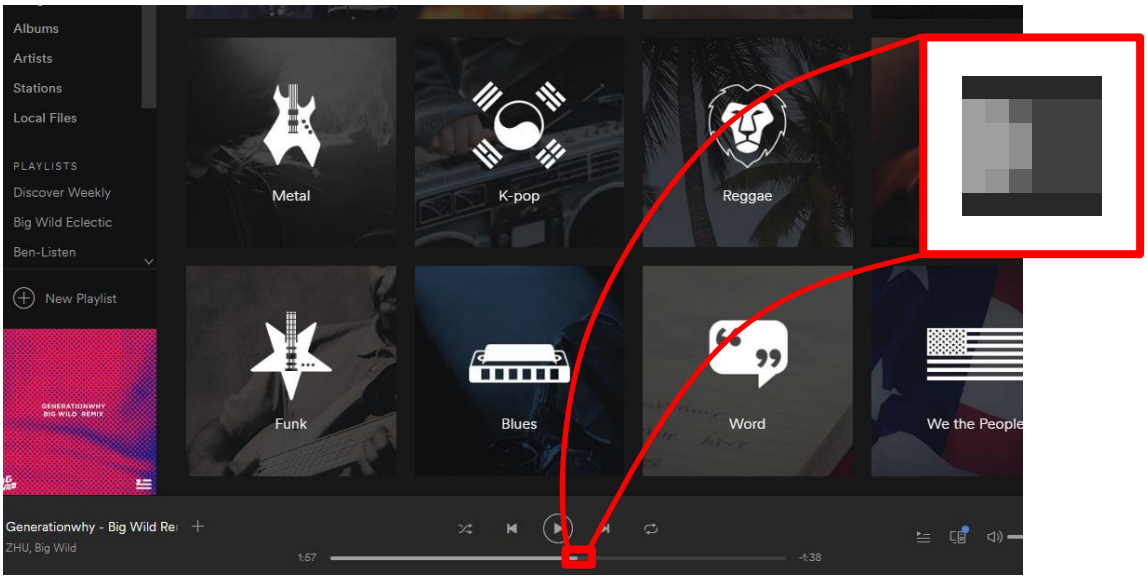
```
^k::FindClick("*0x4A4A4A 4x4", "x5 y5 k{RButton}d r a5,20,-100,15")
; 0xF2F2F2 is the color of the active tab with the default theme.
Replace this with whatever color your theme has.
; k{RButton}d = when it is found, instead of clicking right click and
then press d to select "duplicate" in the menu
; a5,20,-100,15 = only seeks the top sliver of the chrome window
```

SEEK IN SPOTIFY

Uses Up, Down, Left, Right keys to seek back in forth in the current track in Spotify for desktop.

```
#IfWinActive ahk_class SpotifyMainWindow
~$Right:: ; ~ hotkeys are used so arrow key navigating will still work
~$Left::
~$Up::
~$Down::
SeekShort = 30 ; will move this many pixels with left/right
SeekLong = 100 ; will move this many pixels with up/down
If !FindClick("spotify_seek", "r a250,-34,-250,25 x"
(RegExMatch(A_ThisHotkey, "Right|Up") ? "+" : "-")
(RegExMatch(A_ThisHotkey, "Up|Down") ? SeekLong : SeekShort))
    SoundPlay, %A_Windir%\Media\Windows Ding.wav ; If the image isn't
found then play the windows "ding" sound indicating an error.
Return
/* Explanation of FindClick parameters:
r = relative to the active (Spotify) window
a250,-34,-250,25 = approximate area within the window where the seekbar
    will be found. This isn't strictly necessary but improves
    performance. The search area has the following characteristics:
    250 – left edge is 250px from the left edge of the window
    -34 – top edge is 34px from from the bottom edge of the window
    -250 – right edge is 250px from the right edge of the window
    25 – rectangle is 25px tall
x(ternary expression) = will click either to the left (negative x) or
    to the right (positive x) of the seekbar based on which hotkey
    is pressed.
*/
```

The image segment used can be seen below. Note that it is only 6x6 pixels.



Using an image like this may not always work – the scrollbar could have been a vector image that changed as the song plays. But it's worth trying with FindClick, and in this case it works just fine regardless of seek position.

FREQUENTLY ASKED QUESTIONS

BUTTONS THAT LOOK DIFFERENT WHEN THE MOUSE HOVERS

What if I want to use FindClick to click on some button, but when I use the screenshot creator and hover over the button it looks different?

To work around this issue you need to check the box that says “Allow Offset” in the screenshot creator GUI. When you use this setting, the magnification box will move relative to where it was left when the script was last paused. This means you can pause the script, move the mouse, and then unpause the script so that the magnification area will not be right underneath the mouse, and you will be able to magnify the button as it looks without the mouse hovering over it.

CHOOSING AMONG MULTIPLE RESULTS

For instance, say there are 17 instances of a particular image onscreen that have been detected with the **e** option. How do you choose the 4th from the top?

The answer is you need to write a bit more AutoHotkey code to parse the results. The method for the **e** option can return the results in a somewhat arbitrary order, so telling FindClick you want a particular image number doesn’t mean a whole lot.

```
; The below code stores the coordinate pairs in %Results%
; "n" tells FindClick to not click on the images because you just want
to know where they are
; "fy,x" tells FindClick to format the results with the y-coordinate
first, then a comma, and then the x-coordinate. This will allow sorting
by the y-coordinate
Results := FindClick("image", "e n fy,x")
Sort, Results, N ; Will sort the results based on the y-coordinate. N is
numerical sort
Loop, Parse, Results, `n ; Parse the list one coordinate pair at a time
    If (A_Index = 4) { ; Stop at the 4th item
            StringSplit, Coords, A_LoopField, `, ; Split by comma
            MsgBox, The coordinates are (%Coords2%,%Coords1%) ; Reverse
the order so it becomes x,y again
            Break
    }
```

If you want to sort based on the x-coordinate, you can omit a few of the steps above.

```
Results := FindClick("image", "e n")
Sort, Results, N ; x-coordinate is first by default so will determine
the sorting order
Loop, Parse, Results, `n
    If (A_Index = 4) {
            MsgBox, The coordinates are (%A_LoopField%)
            Break
```

```
    }
```

SPECIAL THANKS

Tic + Rseding91 for Gdip.ahk
http://www.autohotkey.com/board/topic/29449-gdi-standard-library-145-by-tic/?p=533310

Chris + Lexikos for AutoHotkey
http://www.autohotkey.com/