

GDIplus Canvas

This is [GDIP_Canvas.au3](#) include for AutoIt 3 (at the writing time : version v3.3.18.0).
It aims to make the use some of the GDIPlus functions easier.
It is an "Include and Forget" solution which releases all used resources automatically
(as long as these were created by the commands from this UDF).

p.s. The resources that you load or use outside the commands from this script, you have to release manually.

[!Gui_Template.au3](#) shows the basic functioning of this Include.

The idea is to open a Canvas and to perform the drawing operations on it using simplified commands:

```
#include "GDIP_Canvas.au3"  
#NoTrayIcon
```

```
Window("Gui", 640, 480, 0, 30)  
GUISetBkColor(0x606060, $hgui)  
Canvas(640, 480, 0, 0)
```

```
While GUIGetMsg() <> -3 ;or use $GUI_EVENT_CLOSE  
    ;Cls()  
    Color(Rand(80, 0xffffffff))  
    Print("Hello World", Rand(0, 600), Rand(0, 400))  
    Flip(15)  
Wend
```

p.s. the Canvas is displayed only when the Flip() command is called.

(p.p.s. The standard gui, by using GuiGetMsg() can slow down some programs, in some cases.
if you move the mouse and the drawing speeds up, then its because of the the GuiGetMsg())

There are 2 alternative templates which can be used in a different kind of scenarios.

[!GuiOnEvent_Template.au3](#) uses the AutoIt's Gui On Event option. The advantage of this template is that the GUI can be closed when the script is in a tight loop.

Please see AutoIt's help file or [Forum](#) on how to program in the GuiOnEvent style.

[!GuiGetMsg-ReplacementTemplate.au3](#) uses an alternative to the GuiGetMsg approach, but all handling of buttons and other things need to be done inside the MY_WM_COMMAND and MY_SYS_COMMAND functions. Please see the provided [example](#).

[!GuiOnEvent_TransparentWindow_Template.au3](#) uses GuiOnEvent option and has a basic setup to display [Transparent images](#) over the desktop.

Information:

All drawing operations that you want to see, should be done on the Canvas buffer.

The active buffer can be switched by the `SetBuffer()` command.

To use a Loaded or Created image for the drawing operation use `SetBuffer(ImageBuffer($image))`.

After you are done with the drawings on this buffer, switch it back to the backbuffer by using `SetBuffer(BackBuffer())`.

If you want to display the changes made on the Image buffer, you have to draw in on the Canvas buffer and to use the `Flip()` command to display it to the Gui.

The Canvas can be positioned and moved/resized to anywhere on the Gui with the `SetCanvas`.

You can get 3 types of Mouse positions: use `GetMouseCoords()` once in a loop then the `MouseX()`, `MouseY()`, `MouseGX()`, `MouseGY()`, `MouseRX()` and `MouseRY()` will contain the updated coordinates.

`MouseX/Y` holds the coordinates for the Canvas. It takes in account if the canvas is smaller or bigger, and may skip some points if you display for e.g. 640x480 as a 320x240 Canvas.

`MouseRx,Y` holds the relative coordinate desktop to the window position.

`MouseGx,Y` holds the Global desktop coordinates.

`BlendMode` command may or may not work, depending on the cases. It is using the `_GDIPlus_GraphicsSetCompositingMode()` function. Theoretically it should draw the transparent bits of a Picture as non-transparent. It affects other operations as well.

Color and Background colors are set up and used until changed = Calling the `Color()` and `Cls()` without parameters will reuse the previous colors.

`Print` command displays the text by pre-defining the Font settings with `LoadFont`. You can have up to 20 fonts loaded into the slot numbers and the slots can be chosen with `SetFont(nr)`.

Loading a font over an existing font slot will free the previously used data.

`Text` command, on the other hand, does not need a pre-defined font, but you can set up the font directly.

You can have up to 255 images and 20 Font slots. (unless you change it in the `GDIP_Canvas.au3` if needed)

Window style can be set up only before the Window is opened. The commands for `SetGuiStyle` and `SetGuiExStyle` are useless afterwards. Please see the `AutoIt` help file for the Gui styles.

`Circle` and `Ellipse` are defined from top left hand corner with width and height.

`Polygon` needs 3 pairs of coordinates passed as a string. the x,y coordinate pair is separated by the semicolon ; .

There is a [Polygon editor](#) which can help with the drawings.

Commands / Functions List:

Color
RGB
GetColor
PenSize
Plot
Line
Rect
Circle
Ellipse
Pie
Polygon
DrawImage
DrawImageSize
DrawImageZoom
DrawAnimImage
SetAnimImage
DrawImageFlip
DrawRotate
BlendMode
Flip
Cls
SetBuffer
BackBuffer
ImageBuffer
SaveImage
LoadImage
CreateImage
CloneImage
ScreenShotImage
FreeImage
ImageToClipboard
CanvasToClipboard
ClipboardToImage
AppTitle
ImageWidth
ImageHeight
AnimWidth
AnimHeight
CanvasWidth
CanvasHeight
CanvasDrawWidth
CanvasDrawHeight
Rand
Text

Print
SetFont
FontStyle
LoadFont
FreeFont
FontSize
FontName
MouseX
MouseY
MouseRX
MouseRY
MouseGX
MouseGY
MouseZ
GetMouseCoords
KeyDown
Window
Canvas
SetCanvas
SetGuiStyle
SetGuiExStyle
GuiHWND
Debug
GetSaveNameDialog
GetLoadNameDialog
SetProcessDpiAwareness

Example Scripts (Tools/Games/Graphics):

[AsciiNumberShooter.au3](#)
[BarnsleyFern-Rosettacode.au3](#)
[Clock.au3](#)
[Eyes.au3](#)
[Eyes\(transparent\).au3](#)
[BubbleUniverse.au3](#)
[BubbleUniverse-GuiGetMsg-Replacement.au3](#)
[FloodFill_GoogleAI.au3](#)
[FlyingLetters-GuiOnEvent.au3](#)
[Non-Periodic-Facebook-GuiGetMsg-Replacement.au3](#)
[Non-PeriodicPolar-Facebook-GuiGetMsg-Replacement.au3](#)
[Pendulum-Rosettacode.au3](#)
[PolygonEditor.au3](#)
[Ripple-Anim-GuiGetMsg-Replacement.au3](#)
[Stars+FPS+GuiOnEvent.au3](#)
[StarsV2_Transparent.au3](#)

Commands Explanation:

p.s. \$hImage is a pointer to the loaded/created image not to the actual GDI-Handle. (mostly)

Color (\$color = 0x000000, \$alpha = 0xff)

[Open Example](#)

Sets the drawing color in RGB format. The color will be used until changed.
Decimal and Hexadecimal formats can be used.

Example:

```
Color (0xFF0000)
Color (0x00FF00)
Color (0x0000FF)
```

To change only the alpha value, set the color value to -1
Alpha values are in between 0 and 255

```
Color (-1, 0xFF)
Color (-1, 255)
Color (-1, 30)
```

The RGB function is created for this case.

RGB (\$R, \$g, \$b)

[Open Example](#)

Combines 3 values into a value useable by the color function.

R,G,B can be in the range of 0 to 255 (0x0 to 0xFF)
Use this function to set the Rgb values in the color function.

Example:

```
Color (RGB(255,255,255))
Color (RGB(0x11,0xFF,0x3F))
Cls (RGB(255,0,0))
```

GetColor (\$x, \$y)

[Open Example](#)

Returns the Pixel color from the current buffer at the specified coordinates.
The returned value is in decimal. The alpha value is saved in **\$GetColorAlpha**
p.s. Clear the canvas with Cls(-1,0) to get the alpha color values, or you will get the wrong alpha values.

Example:

```
$c=GetColor(10,30)
```

PenSize (\$size = 1)[Open Example](#)

Sets the size of the Pen for the drawing functions that supports them.

(Line, Rect, Circle, Ellipse, Pie, Polygon)
p.s. works only in non filled mode.

Plot (\$x, \$y, \$size = 1)[Open Example](#)

Draws a pixel on the current buffer. Increased size makes a filled Square.

Line (\$x, \$y, \$x1, \$y1)[Open Example](#)

Draws a Line at the specified coordinates.

Rect (\$x, \$y, \$w, \$h, \$f = 0)[Open Example](#)

Draws a Rectangle/Square at the specified coordinates with Width and Height as size
\$f = 1 creates a filled Rectangle

Circle (\$x, \$y, \$s, \$f = 0)[Open Example](#)

Draws a circle at the x,y coordinates with S as size
\$f = 1 creates a filled Circle
p.s. X,Y coordinate are at the top left side of the Circle

Ellipse (\$x, \$y, \$sx, \$sy, \$f = 0)[Open Example](#)

Draws a Ellipse at the x,y coordinates with SX and SY as size
\$f = 1 creates a filled Ellipse
p.s. X,Y coordinate are at the top left side of the Ellipse

Pie (\$x, \$y, \$w, \$h, \$a1, \$a2, \$f = 0)[Open Example](#)

Draws a Pie at the x,y coordinates with Width and Height as size
a1 The angle, in degrees, between the X axis and the starting point of the arc that defines the pie. A positive value specifies clockwise rotation.
a2 Same as a1 but for the End point.
\$f = 1 creates a filled Pie
p.s. X,Y coordinate are at the top left side of the Pie

Polygon (\$f = 0, \$str = "10,10;60,10;30,30;10,10")

[Open Example](#)

Draws a polygon. f 0/1 determines if it is filled or not.
This polygon type is Fixed, it is not freely moveable.
Str needs to have a coordinate pairs, in X,Y format, in a STRING
The coordinates are separated by ;
You need at least 3 pairs of coordinates.
The polygon needs to have Start and end points at the same coordinates.
If not, an ending line between start and end will be automatically added. (by the GDIP)

DrawImage (\$hImage, \$x, \$y)

[Open Example](#)

Draws a loaded or created image at the x,y position without any scaling.

DrawImageSize (\$hImage, \$x, \$y, \$w, \$h)

[Open Example](#)

Draws a loaded or created image at the X,Y position with Width and Height as size

DrawImageZoom (\$hImage, \$sx, \$sy, \$sw, \$sh, \$dx, \$dy, \$dw, \$dh)

[Open Example](#)

Copies a part of loaded or created image from a starting position SX,SY and SW,SH (Width/Height)
and draws it at the DX,DY position with DW,DH (Width/Height) as size

DrawAnimImage (\$hImage, \$x, \$y, \$w, \$h, \$frame = 1, \$flip = 0)

[Open Example](#)

Draws an animated image at the position X,Y with W,H (Width/Height)
with frame as numbers in between 1 and xxx as pre-calculated by the SetAnimImage
The image can be Flipped/Mirrored (see DrawImageFlip for the numbers)

SetAnimImage (\$hImage, \$w, \$h, \$fw, \$fh)

[Open Example](#)

Precalculates an image to be used with DrawAnimImage.
W,H determines the size of the single frame
Fw,Fh = Frame count Horizontally/Vertically

p.s. there is no error checking.
Image frames are positioned as in this example (fw=4 and fh=3):

1	2	3	4
5	6	7	8
9	10	11	12

DrawImageFlip (\$hImage, \$x, \$y, \$w, \$h, \$flip)

[Open Example](#)

Draws an image to X,Y coordinates with W,H (Width/Height) size flipped or mirrored.

\$a determines the rotation which can be one of this:

- 0 - No rotation and no flipping (A 180-degree rotation, a horizontal flip and then a vertical flip)
- 1 - A 90-degree rotation without flipping (A 270-degree rotation, a horizontal flip and then a vertical flip)
- 2 - A 180-degree rotation without flipping
(No rotation, a horizontal flip followed by a vertical flip)
- 3 - A 270-degree rotation without flipping
(A 90-degree rotation, a horiz. flip and then a vertical flip)
- 4 - No rotation and a horizontal flip (A 180-degree rotation followed by a vertical flip)
- 5 - A 90-degree rotation followed by a horizontal flip (A 270-degree rotation with a vertical flip)
- 6 - A 180-degree rotation followed by a horizontal flip (No rotation and a vertical flip)
- 7 - A 270-degree rotation followed by a horizontal flip (A 90-degree rotation with a vertical flip)

DrawRotate (\$hImage, \$iX, \$iY, \$w, \$h, \$fAngle, \$flip)

[Open Example](#)

Draws an X,Y coordinates with W,H (Width/Height) size and freely rotated with fAngle
Negative angle numbers will rotate it Anti-Clockwise. \$flip flips the image
See DrawImageFlip for the explanation.

BlendMode (\$blend)

[Open Example](#)

[Experimental](#) - may or may not work, depending on the case.

Turns the blending mode off or on (0/1) for the follow up commands (until changed)
(the transparent bits of an image are drawn in solid color when turned on)

Flip (\$fps = 50, \$sleep = 0)

[Open Example](#)

This is needed for the canvas to be drawn at all. Standard fps value is 50.

fps is optional and delays the frame rate down to the set fps value.

p.s. It is a simple frame limiter for the faster machines, attempting to hold the set fps rate.
(may or may not be accurate). When \$sleep is set to 1 it adds a sleep (10) to each waiting loop, but only if the frame rate is achieved. Calls a GetMouseCoords() as well.

There is a **known bug**: Flip() does not show anything when it is called the first time.

Cls (\$c = -1, \$a = -1)

[Open Example](#)

Clears the current (active) buffer in the Color and Alpha values.

Cls remembers the values so that the next calls without parameters will use the same color settings.

Example:

Cls (0x0)

Cls (RGB(0xFF,255,255))

SetBuffer (\$b)

[Open Example](#)

Changes a buffer to be used. all follow up commands will perform their operation on it.

Usage example: SetBuffer (BackBuffer())
SetBuffer (ImageBuffer(\$hImage))

BackBuffer ()

[Open Example](#)

Returns the handle of the BackBuffer.
Used in combination with SetBuffer()

ImageBuffer (\$hImage)

[Open Example](#)

Returns the handle of one of the Images
Used in combination with SetBuffer()

SaveImage (\$hImage, \$FileName, \$direct = 0)

[Open Example](#)

Saves an Image under the FileName path.
if direct is set to 1, you can actually pass a custom loaded image handle instead of the built in handle.

Caution: This will **overwrite** any existing file, or **may not** save if the file is locked.

LoadImage (\$sImage)

[Open Example](#)

Loads an image into the ImageArray and returns the handle.

Remember to always free an image before loading another into the same variable.

CreateImage (\$w, \$h)

[Open Example](#)

Creates an empty image with W/H (Width/Height) as size

CloneImage (\$hImage, \$direct = 0)

[Open Example](#)

Creates a copy of an existing image and returns the handle to it.
direct 1 is used to copy custom loaded images.

ScreenShotImage(\$x = 0, \$y = 0, \$x1 = -1, \$y1 = -1, \$hide = 0)

[Open Example](#)

Takes a screenshot and Creates an image.
if \$x1 or \$y1 are set to -1 then the Desktop Width/Height will be used
if \$hide is set to 1, the Gui Window will be hidden before the capture.

Important: Multi monitor setups with different scaling sizes may take wrong screenshots
Use the **SetProcessDpiAwareness(2)** at the top of your Screenshot taking script.

FreeImage (\$hImage, \$direct = 0)[Open Example](#)

Releases the used resources.

direct is used to release custom loaded image with `_GDIPplus_BitmapDispose($hImage)`

ImageToClipboard (\$hImage, \$direct = 0)[Open Example](#)

Copies an Image to the Clipboard.

direct is used to copy a custom loaded image. (untested)

CanvasToClipboard ()[Open Example](#)

Copies the current Canvas image into Clipboard.

ClipboardToImage ()[Open Example](#)

Copies the content of the Clipboard into one image slot and returns a handle to the image.

Example:

```
$clipimage = ClipboardToImage()  
DrawImage ($clipimage,0,0)
```

AppTitle (\$newtitle = "GDIP Canvas")[Open Example](#)

Sets a new app title

Example:

```
AppTitle("Hello World")
```

ImageWidth (\$hImage, \$direct = 0)[Open Example](#)

Returns the width value of an loaded/created image.

direct is used for the custom loaded images

```
$image=LoadImage("test.png")  
$iw=ImageWidth($image)
```

ImageHeight (\$hImage, \$direct = 0)[Open Example](#)

Returns the Height value of an loaded/created image.

direct is used for the custom loaded images

```
$ih=ImageHeight($image)
```

AnimWidth (\$hImage)[Open Example](#)

Returns the Width of a previously created AnimImage \$hImage

AnimHeight (\$hImage)[Open Example](#)

Returns the Height of a previously created AnimImage \$hImage

CanvasWidth ()[Open Example](#)

Returns the total Width of the Canvas

CanvasHeight ()[Open Example](#)

Returns the Total Height of the Canvas

CanvasDrawWidth ()[Open Example](#)

Returns the drawn Width of the Canvas

CanvasDrawHeight ()[Open Example](#)

Returns the drawn Height of the Canvas

Rand (\$min = 0, \$max = 1)[Open Example](#)

Returns a random integer numbers in range of min, max

Text (\$txt, \$x, \$y, \$font = "Arial", \$size = 12, \$format = 0x0)[Open Example](#)

Draws a Text at X,Y coordinates with Custom defined Font, size and format bypassing the built in font.

Format flags can be one or more of the following. Default = 0 :

0x0001 - Reading order is right to left

0x0002 - Individual lines of text are drawn vertically on the display device

0x0004 - Parts of characters are allowed to overhang the string's layout rectangle

0x0020 - Unicode layout control characters are displayed with a representative character

0x0400 - An alternate font is used for characters that are not supported in the requested font

0x0800 - The space at the end of each line is included in a string measurement

0x1000 - The wrapping of text to the next line is disabled

0x2000 - Only entire lines are laid out in the layout rectangle

0x4000 - Characters overhanging the layout rectangle and text extending outside the layout rectangle are allowed to show

Example:

Text ("A U T O IT", 20, 30, "Arial", 12, BitOr(0x1,0x2,0x4, 0x20))

Print (\$txt, \$x, \$y)

[Open Example](#)

Draws a text at X,Y coordinates.
See SetFont() and LoadFont() commands

Example:

```
LoadFont(1,"Arial", 9, FontStyle(0,1,0,0))
SetFont (1)
Print ("Hello", 3,10)
```

SetFont (\$nr)

[Open Example](#)

Changes a font to a previously loaded font. (p.s. It ignores any unused numbers.)
In multiple font settings, if a font is released and switched to the freed number, it will not change the font and previous font may be used.
Nr can be in range of 1-20 (by default) ... unless it was changed in the GDIP_Canvas.au3

FontStyle (\$bold = 0, \$Italic = 0, \$Underline = 0, \$Strikethrough = 0)

[Open Example](#)

Sets a font style flags (or combination of them).
Used with the LoadFont () command

Example

```
LoadFont(1,"Arial", 9, FontStyle(0,1,1,0))
```

LoadFont (\$nr, \$FontName = "Arial", \$size = 12, \$style = 0)

[Open Example](#)

\$nr has to be in range of 1 to Ubound(\$CanvasINC_Font,1)-1
Font name has to be a valid font family name like "Arial" or "Consolas"
Font size may or may not work for every font.
The style of the typeface can be a combination of the following
(or use the FontStyle() command) :

- 0 - Normal weight or thickness of the typeface
- 1 - Bold typeface
- 2 - Italic typeface
- 4 - Underline
- 8 - Strikethrough

Example:

```
LoadFont(1,"Consolas", 11, 3)
```

FreeFont (\$nr)

[Open Example](#)

Releases the font resources used for the Print command.
If you release an currently used font number, the print command will display nothing.

FontSize ()[Open Example](#)

Returns the current font Size

FontName ()[Open Example](#)

Returns the current font Name.

MouseX ()[Open Example](#)

Returns the MouseX coordinates when the GetMouseCoords() was used, previously
This is Canvas (size) based X coordinate.

Example:

```
While 1
    GetMouseCoords()
    $mx=MouseX()
    Print ($mx,0,0)
    Flip()
Wend
```

MouseY ()[Open Example](#)

Returns the MouseY coordinates when the GetMouseCoords() was used, previously.
This is Canvas (size) based Y coordinate.

Example:

```
While 1
    GetMouseCoords()
    $my=MouseY()
    Print ($my,0,0)
    Flip()
Wend
```

MouseRX ()[Open Example](#)

Returns the MouseRY coordinates when the GetMouseCoords() was used, previously
This is Global but Relative to the client area X coordinate.

MouseRY ()[Open Example](#)

Returns the MouseRY coordinates when the GetMouseCoords() was used, previously
This is Global but Relative to the client area Y coordinate.

MouseGX ()[Open Example](#)

Returns the MouseGY coordinates when the GetMouseCoords() was used, previously
This is Global (desktop) X mouse coordinate.

MouseGY ()[Open Example](#)

Returns the MouseGY coordinates when the GetMouseCoords() was used, previously
This is Global (desktop) Y mouse coordinate.

MouseZ ()[Open Example](#)

Returns the Mouse Wheel position.

Example:

```
Global $TMPMW="" , $tmp , $x=0

While GUIGetMsg() < -3
    cls()

    $TMPMW=MouseZ()
    $x=$x+$TMPMW

    if $TMPMW=1 then
        $tmp="Up"
    ElseIf $TMPMW=-1 Then
        $tmp="Down"
    EndIf

    Print ("Mouse Wheel is moving " & $tmp,0,0)
    Print ($x,0,20)
    Flip(20)
Wend
```

GetMouseCoords ()[Open Example](#)

Use this function once in the main loop to get the mouse coordinates
(p.s. Flip() is calling this function as well. If you are using flip then you do not need to use this again. It is added for the cases where you might want to get the coordinates without using the canvas at all.)

KeyDown (\$key = "")[Open Example](#)

Available keys:

Up, Down, Left, Right, Space, Ctrl, Alt, Shift, Enter,
Lmouse, Rmouse, Mmouse,
LShift, RShift, Lctrl, Rctrl, ESC
F1-F12

Example:

```
If Keydown("up") then Print ("Arrow up",0,0)
If Keydown("left") then Print ("Arrow Left",0,20)
If Keydown("Mmouse") then Print ("Middle Mouse Button",0,40)
If Keydown("F1") then Print ("F1 Pressed",0,60)
```

Window (\$title = "Canvas", \$WW = 320, \$WH = 240, \$PX = -1, \$PY = -1)[Open Example](#)

Opens the Gui Window for use with the canvas.
Canvas command is opening this automatically, if the gui window is not open.

Example:

```
Window ("Example 01", 600,300)
```

Canvas (\$width = 320, \$height = 240, \$PosX = 0, \$PosY = 0)[Open Example](#)

Creates the canvas with the Specified Width and Height and optionally sets the drawing place which defaults to the 0,0 position.
Canvas will call the Window function automatically, if it is not open. The window will have the Width and the Height of the Canvas.

Example:

```
Canvas (320,320,0,30)
```

SetCanvas (\$x = -1, \$y = -1, \$w = -1, \$h = -1, \$ZoomOff = 0)[Open Example](#)

Sets (changes) the Canvas to a new Coordinate or to a new Size.
If the opened canvas Width and Height are smaller than this command Width/Height, the Canvas will be displayed as zoomed in or Zoomed out (if smaller).
ZoomOff when set to 1 will resize the Canvas to the new size.
If the new canvas size is bigger than the Created size, the canvas will be automatically Resized. This may cause a loss of whatever was drawn on it, previously.

Example:

```
Window("Scaled up Canvas",640,320)
Canvas(320,160) ;Small sized canvas
SetCanvas (0,0,640,320) ;Creates a Canvas scaled up to the Window size.
```

SetGuiStyle (\$style = BitOR(\$WS_SIZEBOX, \$WS_CAPTION))

[Open Example](#)

Sets a Style for the window. This **needs** to be called Before the window is opened.

Example:

```
SetGuiStyle ($WS_POPUP)
Window ("Popup window style",640,480,0,30)
```

SetGuiExStyle (\$style = \$WS_EX_WINDOWEDGE)

[Open Example](#)

Sets an Extended Style for the window. This **needs** to be called Before the window is opened.

Example:

```
SetGuiExStyle (BitOR($WS_EX_LAYERED, $WS_EX_TOPMOST))
Window ("Layered window EX style",640,480,0,30)
```

GuiHWND ()

Returns the handle of the window, which is alternatively saved in the \$hGui

Debug (\$on = 1)

Turns the debugging information on/off. (0 or 1)
Not every command has debugging information available.
The error information will be displayed in the Scite console.
The error codes numbers are listed at the **bottom** of this document.

GetLoadNameDialog(\$file = @ScriptDir, \$title = "Open Image", \$styp = 1)

[Open Example](#)

Opens an File Open dialog and returns the filename.

\$file should contain the folder to be opened.
\$styp 1 will set the filter to image files (png, jpg, bmp, gif, tiff)
\$styp 0 will set the filter to All Files (*.*)

Example usage:

```
$fname = GetLoadNameDialog()
If @error=0 Then
    ConsoleWrite("Filename: " & $fname & @crlf)
Else
    ConsoleWrite("Invalid filename" & $fname & @crlf)
EndIf
```

p.s. always check if @error is 0 because the returned filename is the same as \$file, if the selection was invalid or cancelled.

GetSaveNameDialog(\$file = @ScriptDir, \$title = "Save Image", \$typ = 1)

[Open Example](#)

Opens an File Save dialog, and returns a valid Filename (with extension)

\$file should contain the folder to be opened.

\$typ 1 will set the filter to image files (png, jpg, bmp, gif, tiff)
the extension will be added for the selected image type.

\$typ 0 will set the filter to All Files (*.*)
the extension will not be added here.

If a filename exists, the user will be asked to overwrite the file or to change the name.

Example usage:

```
$fname = @scriptdir & "test.png"
$fname = GetSaveNameDialog($fname)
If @error=0 Then
    ConsoleWrite("Filename: " & $fname & @crlf)
Else
    ConsoleWrite("Invalid filename" & $fname & @crlf)
EndIf
```

p.s. always check if @error is 0 because the returned filename is the same as \$file, if the selection was invalid or cancelled.

p.p.s this does not check if the place has writing permissions.

SetProcessDpiAwareness(\$DPIAware=2)

[Open Example](#)

Sets the DpiAwareness for the Current App.

This is **needed** for taking Screenshots on differently scaled (multi-monitor) setups !
Use it at the beginning of your Script with the default value of 2

Valid arguments are:

```
$PROCESS_DPI_UNAWARE           = 0
$PROCESS_SYSTEM_DPI_AWARE      = 1
$PROCESS_PER_MONITOR_DPI_AWARE = 2
```

Helper Function names:

P.S. (These are internally used by the script.)

General Helper functions:

DebugWrite (\$txt)

GDIP_GameUDF_CombineColors (\$a, \$c)

_Bitmap2Clipboard (\$hImage)

_WinSetClientPos (\$sTitle, \$sText, \$w, \$h, \$l = Default, \$t = Default)

_Mousewheel (\$hWnd, \$iMsg, \$wParam, \$lParam)

GDIP_GameUDF_Close ()

AutoIt options:

AutoItSetOption ("MouseCoordMode", 1 and 2) used in the GetMouse() function

AutoIt Message handlers:

OnAutoItExitRegister ("GDIP_GameUDF_Close")

GUIRegisterMsg (\$WM_MOUSEWHEEL, "_Mousewheel")

Used Variable names:

GDIP_Canvas uses the variables starting with \$CanvasINC for its internal functioning with the exception of \$hGui and \$GuiGetAlpha

A copy of GDI Error Codes when the Debug is enabled:

(from the AutoIt GdiPlus include file)

ERROK = 0 ; Method call was successful
ERRGENERICERROR = 1 ; Generic method call error
ERRINVALIDPARAMETER = 2 ; One of the arguments passed to the method was not valid
ERROUTOFMEMORY = 3 ; The operating system is out of memory
ERROBJECTBUSY = 4 ; One of the arguments in the call is already in use
ERRINSUFFICIENTBUFFER = 5 ; A buffer is not large enough
ERRNOTIMPLEMENTED = 6 ; The method is not implemented
ERRWIN32ERROR = 7 ; The method generated a Microsoft Win32 error
ERRWRONGSTATE = 8 ; The object is in an invalid state to satisfy the API call
ERRABORTED = 9 ; The method was aborted
ERRFILENOTFOUND = 10 ; The specified image file or metafile cannot be found
ERRVALUEOVERFLOW = 11 ; The method produced a numeric overflow
ERRACCESSDENIED = 12 ; A write operation is not allowed on the specified file
ERRUNKNOWNIMAGEFORMAT = 13 ; The specified image file format is not known
ERRFONTFAMILYNOTFOUND = 14 ; The specified font family cannot be found
ERRFONTSTYLENOTFOUND = 15 ; The specified style is not available for this font
ERRNOTTRUETYPEFONT = 16 ; The font retrieved is not a TrueType font
ERRUNSUPPORTEDGDIDIVERSION = 17 ; The installed GDI+ version is incompatible
ERRGDIPLUSNOTINITIALIZED = 18 ; The GDI+ API is not in an initialized state
ERRPROPERTYNOTFOUND = 19 ; The specified property does not exist in the image
ERRPROPERTYNOTSUPPORTED = 20 ; The specified property is not supported