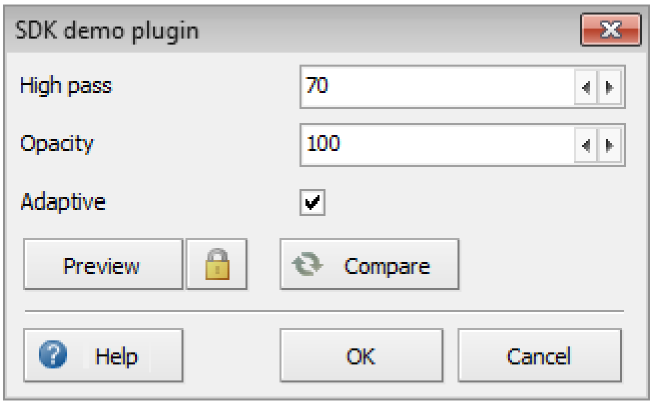
Astroart SDK - Filter plug-ins

Version 8.01 - 2024/02/12

Introduction

A "plug-in" is an external module (DLL) that extends an application. For a software like Astroart plug-ins are related to image processing, file management or astronomical calculations. This part of the SDK refers only to image filters, which use the Astroart Dialog Window:



High pass 70

Opacity 100

Adaptive ☑

Astroart will manage automatically the preview, the real-time preview, the comparison with the original image, the undo and the guide. The programmer only need to code the filter itself.

if instead your goal is to create a generic plug-in, with your own dialog windows, use the "Generic" SDK, which provides more flexibility.

Requirements

This part of the SDK is compatible with Astroart 8.0 SP5 or newer, 32 and 64 bits.

Since Astroart plug-ins are standard DLLs, any compiler can be used for this purpose. This SDK contains some examples in C++ and Pascal (Visual Studio, CodeBlocks GCC, Lazarus, Delphi). Although almost every programming language allows creation of DLLs, this is often an uncommon task for most programmers. We recommend to take a look at the specific documentation of your compiler.

Let's start

Take a look and compile the demo included in this archive using your favorite compiler. Then copy the DLL into the Astroart installation folder. In Astroart open an image, then launch the example from the menu "plug-in".

Astroart looks for plug-ins when it starts. For every plug-in correctly recognized there will be a new command in the Plug-ins Menu. The name of the DLL must be compatible with the following convention:

32 bit: PI*.DLL 64 bit: P64I*.DLL

Example of valid plug-in names: PIFFT.DLL , PIGAUSS.DLL , P64IFFT.DLL , P64IGAUSS.DLL .

Exported functions

A DLL (dynamic link library) is a collection of functions which are exported to applications. Astroart needs four functions from a filter plug-in:

pi_initialize, **pi_dialog**, **pi_process**, **pi_finalize**.

Note the calling convention: WINAPI (STDCALL) this is the standard for Windows. The C++ declaration uses void* to indicate generic pointers, this can be safely changed.

pi_initialize

This functions is called when Astroart starts and loads for plug-ins.

C++ declaration:

```
int WINAPI pi_initialize(void *funcaddress, char *menuname, char *description, HWND whandle)
```

Generic declaration:

```
function pi_initialize (int32/64, int32/64, int32/64, int32/64) -> int32
```

menuname and *description* are pointers to buffers where the function should write the title of the menu and the hint which appears on the status bar of Astroart. Example (C++):

```
strcpy(menuname, "Coma correction")
```

```
strcpy(description, "A plug-in for coma correction by J. Brown")
```

whandle is the Window Handle of the Astroart main window, it can be ignored. *funcaddress* is a pointer to a special function of Astroart; it must be stored in a global variable for future use, see *types & commands* below.

If some initialization is required in your code, it's usually better to do that at the first process, at the function *pi_process* below, not here. This function must return a non-zero value if it executes correctly.

pi_dialog

This function is called when the user clicks the menu item. The plug-in must create here the dialog window.

C++ declaration:

```
void WINAPI pi_dialog(void *pimage, HWND whandle)
```

Generic declaration:

```
procedure pi_dialog (int32/64, int32/64)
```

pimage is the pointer of the current active image, see *types & commands* below, usually it can be ignored.

whandle is the Windows handle of the active MDI Child window, usually it can be ignored.

The plugin can analyze the image (if really needed), then it must create the dialog window, using the command *ac_showdialog*.

```
fcallback(ac_showdialog, DLG_NAME, parameters, flags)
```

The *parameters* are encoded as a multiline text, for example the dialog in the first page was created with:

```
NUMBER "High pass" 70 0 100 5
NUMBER "Opacity" 70 0 100 5
BOOLEAN "Adaptive" 0
HELP "HighPassDemo.htm"
```

The first field is the *type* (NUMBER, BOOLEAN, STRING, HELP, LIST), the second is the default value. Numeric fields may have a minimum, maximum and increment. Strings must be quoted, for example:

```
STRING "Folder" "C:\Temp\"
```

Lists are displayed as a listbox and a title. It's also possible to indicate the default value. The value is 1-based.

```
LIST "Title" "Option1" "Option2" "Option3" "Option4"  4
```

pi_process

This function is called to process the image.

C++ declaration:

```
void WINAPI pi_process(void *pimage, HWND whandle)
```

Generic declaration:

```
procedure pi_process (int32/64, int32/64)
```

pimage is the pointer of the current active image, see *types & commands* below.

whandle is the Windows handle of the active MDI Child window, it can be ignored.

The first task of the plugin is to retrieve the parameters of the dialog window, using the command *ac_getdialogparam*, for example:

```
pv1 = fcallback(ac_getdialogparam, DLG_NAME, 1, NULL)

pv2 = fcallback(ac_getdialogparam, DLG_NAME, 2, NULL)

pv3 = fcallback(ac_getdialogparam, DLG_NAME, 3, NULL)
```

where parameters are returned as pointers to their values. The plugin can now process the image, using the *undo buffer* and the *image buffer*, both allocated by Astroart. For more information see the demo plugin.

pi_finalize

This function is called when Astroart is going to be closed.

C++ declaration:

```
void WINAPI pi_finalize(HWND whandle)
```

Generic declaration:

```
procedure pi_finalize (int32/64)
```

You may use this function to deallocate memory buffers and resource handlers.

Types and commands

There are two special types that a plug-in must use when interfacing with Astroart: pointer to the *image* and pointer to the *callback function*, which is a special function to be called by the plug-in to perform its tasks.

The callback function

Example of C declaration:

```
typedef int/int64 WINAPI (*TCallBackFunc) (TCommand, void*, void*, void*)
```

Generic declaration:

```
CallbackType = pointer to function (int32, int32/64, int32/64, int32/64) -> int32/64
```

The callback function takes four parameters: the first is the command (example: "Get the image size"), the others are parameters.

Example: callback(ac_getsize, image, &x, &y) will get in x and y the size of the image in pixels.

If the plug-in uses threads for background processing, they should not call the callback function directly. Only the main thread should call the callback function, then pass the parameters to worker threads.

Commands

The first parameter of the callback function is a *command*, defined from the enumeration:

enum AstroartCommand {ac_getbuffer, ac_getsize, ac_redraw, ac_close, ac_savefits, ac_copy, ac_writeheader, ac_readheader, prepareundo, ac_thresholds, ac_transferfunc, ac_getselpoint, ac_getselpoint2, ac_gethwnd, ac_create, ac_open, ac_getimage, ac_getversion, ac_rename, ac_getname, ac_getra, ac_getdec, ac_getx, ac_gety, ac_modified, ac_blink, ac_flip, ac_resize, ac_findcoordinates, ac_reserved_1, ac_selectpoint, ac_selectrect, ac_reserved_2, ac_reserved_3, ac_reserved_4, ac_centeratlas, ac_displayhelp, ac_save8bit, ac_preprocessing, ac_executemacro, ac_showdialog, ac_getdialogparam, ac_getundobuffer}

Command	Parameters and description	Return value
ac_getbuffer ac_getundobuffer	pimage, &buffer, colorplane Gets a pointer to the buffer where the image is stored (32 bit floating point data) and returns 2 if there are no errors. This a fundamental function to modify the image (for example to make filters). If the image has colors then the colorplane parameter specifies the green, red, or blue color plane (R = 2, B = 3, else G). ac_getundobuffer requires Astroart 8 SP4 or newer.	0 , 2
ac_getsize	pimage, &x, &y Gets in X and Y the width and height of pimage. If the function return zero then pimage is an invalid pointer.	0 , 1
ac_redraw	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_close	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_savefits	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_copy	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_writeheader	pimage, pheader, NULL Writes the FITS header of the image from the null-terminated string pheader. If the header contains WCS keywords they will be used for astrometry calibration. Every row of the header must be separated by carriage returns, there is no need for trailing spaces.	0 , 1
ac_readheader	pimage, NULL, NULL Gets the pointer to a null-terminated string representing the FITS header. Every row is separated by carriage returns, this string is read-only.	pheader
ac_prepareundo	Don't use with the Filter SDK. See the Generic SDK.	0 , 1

ac_thresholds	pimage, min, max Sets the minimum and maximum visualization thresholds of the image. Use ac_redraw to update the image.	0 , 1
ac_transferfunc	pimage, trf, NULL Sets the transfer function of the image. Use ac_redraw to update the image. The valid range is [-20..20] , -20 logarithmic, +20 exponential, 0 linear.	0 , 1
ac_getselpoint	pimage, &x, &y Gets the coordinates of the selected point of the image.	0 , 1
ac_getselpoint2	pimage, &x, &y Gets the coordinates of the second selected point of the image, if a rectangular region is selected over the image. If x and y contain the constant 10001H then the function returns the number of points selected over the image. If only x or y contains such constant then the function gets the coordinate (y or x) of the y# or x# point of the image, (0 to N-1).	0 , 1 Count X Y
ac_gethwnd	pimage, NULL, NULL Gets the Window Handle of the active MDI Child window.	HWND
ac_create	Don't use with the Filter SDK. See the Generic SDK.	pimage
ac_open	Don't use with the Filter SDK. See the Generic SDK.	pimage
ac_getimage	Don't use with the Filter SDK. See the Generic SDK.	pimage
ac_getversion	NULL, NULL, NULL Returns the version of the plug-in subsystem. This value is 804 for Astroart 8.0 SP4.	500+
ac_rename	Don't use with the Filter SDK. See the Generic SDK.	pimage
ac_getname	pimage, path, NULL Gets the filename of the image. If path = 1 returns the name and the complete path in the file system.	pname
ac_getra	pimage, x*100, y*100 Returns the R.A. of the pixel at coordinates X,Y for images with astrometric calibration. X and Y are intended as 0.01 pixels. The result is an integer number as degrees*100000.	0 .. 360*100000
ac_getdec	pimage, x*100, y*100 Returns the Declination of the pixel at coordinates X,Y for images with astrometric calibration. X and Y are intended as 0.01 pixels. The result is an integer number as degrees*100000.	-90 *100000 90 * 100000
ac_getx ac_gety	pimage, RA*1E6, DEC*1E6 Given the coordinates RA and DEC (multiplied by 1000000 and rounded) returns the pixel X or Y coordinate for images with astrometric calibration. The result is an integer number expressed as pixel * 100.	X * 100 Y * 100
ac_modified	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_blink	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_flip	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_resize	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_findcoordinates	Don't use with the Filter SDK. See the Generic SDK.	0 , 1

ac_selectpoint	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_selectrect	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_centeratlas	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_preprocessing	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_executemacro	Don't use with the Filter SDK. See the Generic SDK.	0 , 1
ac_showdialog	pname, pparameters, flags Creates a dialog window. The name should be unique for a given dialog. Parameters are encoded as a multiline text, see the paragraph "pi_dialog" above . Flags can be a combination of the following constants: PD_PREVIEW , PD_PREVIEW_REALTIME	0 , 1
ac_getdialogparam	pname, index, NULL Gets the values entered by the user in the dialog window. pname must be the same name used in ac_dialogname, the index is 1-based. The function returns a pointer to the value, (*double* for numeric and boolean values and *null terminated string* for strings). See the demo for more information.	pparam

Technical support

Feel free to contact us, or visit the Astroart web site. Both freeware and commercial plug-ins can be listed in the Astroart web site for free.

MSB software
Via Goetz 93
48124 Ravenna RA - ITALY
TEL +39 0544 527265 [13:00 - 17:00 UT]
CELL +39 339 2739548 [13:00 - 17:00 UT]
EMAIL info@msbsoftware.it
WEB www.msb-astroart.com

History

2024/03/14 LIST parameters for Astroart 8 SP5. Version 8.01

2023/03/14 First release for Astroart 8 SP4. Version 8.0