Astroart SDK - Generic plug-ins

Version 8.0 - 2023/03/14

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. However if your goal is to create an image processing filter, use the "Filter" SDK, which provides easy dialog windows and free previews.

Requirements

This SDK is compatible with Astroart 5.0 / 6.0 / 7.0 / 8.0, 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, C++ Builder, Delphi, Lazarus). 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 plug-in menu.

Astroart looks for plug ins when it starts. For every plug-in correctly recognized there will be a new menu 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 three functions from a valid plug-in:

pi_initialize, **pi_activate**, **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 activation, at the function *pi_activate* below, not here. This function must return a non-zero value if it executes correctly.

pi_activate

This function is called when the user clicks the menu item of to the plug-in.

C++ declaration:

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

Generic declaration:

```
procedure pi_activate (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.

When *pi_activate* is called the plug-in should perform its task. For example a filter should elaborate the active image. If instead the plug-in has a user interface then it should display its dialog window.

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: "Create an image"), the others are parameters.

Example: myimage = callback(ac_create, 400, 300, 0) will create a new gray scale image into Astroart with 400 x 300 pixels, and will return a pointer to the new image for further elaborations.

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	pimage, calcstats, autovis Redraws pimage. This is required after you have modified the image buffer. If calcstats is 1 the Astroart will recalculate the image statistics (recommended). If autovis is 1 then the image will be displayed with a new automatic visualization.	0 , 1
ac_close	pimage, askuser, NULL Closes pimage. If askuser is 1 and the image was modified then a dialog window will ask confirmation to the user.	0 , 1
ac_savefits ac_save8bits	pimage, pfilename, NULL Saves the image to disk. pfilename is the pointer to a null-terminated string for the file name.	0 , 1
ac_copy	pimage, NULL, NULL Copies pimage into the clipboard, as a bitmap.	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	pimage, NULL, NULL Copies the image into an internal Undo buffer. This allows users to undo the operations of the plug-in.	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	x, y, color Creates a new image with a size of x and y pixels. If color is not zero then a color image will be created.	pimage
ac_open	pfilename, NULL, NULL Opens an image from disk, pfilename is null-terminated string. If a I/O error occours it returns NULL.	pimage
ac_getimage	selection, NULL, NULL If selection is the constant PACTIVEIMAGE it returns the active image, if selection is PSELECTIMAGE a dialog window will ask the user to select an image (unless there is only one image in Astroart).	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	pimage, pname, keeppath Renames the image. If path = 1 then the original path of the filename will not be changed.	0 , 1
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	pimage, modified, NULL Sets the modified flag of the image. When an image is marked as modified, it cannot be closed directly (a dialog window asks to save the changes).	0 , 1
ac_blink	NULL , NULL , NULL If two images are opened inside Astroart, they are blinked.	0 , 1
ac_flip	pimage, axis, NULL Flips the image. Axis 1 = horizontal, 2 = vertical, 3 = both.	0 , 1
ac_resize	pimage, dimX, dimY Resizes the image.	0 , 1
ac_findcoordinates	pimage, pparameters, NULL Performs plate solving on the image. A pointer to the following structure must be passed: struct {float imgFovX, float sizeErrorPerc, float maxAngle, float approxRa, float approxDe, float searchRa, float searchDe, float minStars, int isFlipped}. if you set imgFovX = 0.0 (recommended) then imgFovX, sizeErrorPerc and maxAngle will be get from the Astroart settings. If you set searchRa = 0.0 then a default rectangle of 2 x 2 degrees will be used for searching.	0 , 1
ac_selectpoint	pimage, X, Y Selects on the image the point at coordinates X, Y.	0 , 1
ac_selectrect	pimage, prect, NULL Selects on the image a rectangle. A pointer to the following structure must be passed: struct {int x1, int y1, int x2, int y2}.	0 , 1
ac_centeratlas	RA*1E6, DEC*1E6, showatlas Centers the star atlas to the given coordinates. If the atlas is currently not opened you can open it with the flag showAtlas = 1.	0 , 1
ac_preprocessing	pimage, pdetails, NULL Launches preprocessing using the text *pdetails* which contains all the information. Inspect a prepreprocessing file to find out the syntax.	0 , 1
ac_executemacro	pimage, number, NULL Launches one of the 8 macros defined by the user.	0 , 1
ac_showdialog	See the Filter SDK. Usually generic plugins use dialog windows created by the developer. When used in this context (generic plugins) Astroart Dialog Windows will not support the preview and the other advanced features.	0 , 1
ac_getdialogparam	See the Filter SDK.	pparam

Technical support

Feel free to contact us, visit the Astroart web site.

Both freeware and commercial plug-ins will 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

2023/03/14 New features for Astroart 8. Version 8.0

2022/10/27 Updated documentation.

2018/07/14 Support for 64 bits plugins. Version 7.0

2017/11/26 Updated version for Astroart 6. Version 6.0

2013/12/28 First version for Astroart 5. Version 5.0