Plugins Menu

Introduction

Plugins are loadable code modules that extend the capabilities of ImageJ. Plugins located in the plugins folder are listed at the bottom of this menu. Plugins in subfolders of the plugins folder are listed in submenus. Use *Shortcuts/Install Plugin* to install a plugin in a different menu or to assign it a keyboard shortcut. More than 100 plugins are available for download from the ImageJ Web site.

Use the *Record* command to record a series of commands and to convert these commands to a plugin. Use the *New* command to create a new plugin and *Edit* to make changes to an existing one. ImageJ comes with two sample plugins: Inverter, which inverts 8-bit images, and Red and Blue, which generates an RGB image. A tutorial for plugin writers is available at http://mtd.fh-hagenberg.at/depot/imaging/imagej/.

Internal Plugins

Most commands in ImageJ are implemented as plugins but these internal plugins are located in the ij.jar file, not in the plugins folder. ij.jar also contains the properties file (IJ_Props.txt) that ImageJ uses to install internal plugins in menus. A JAR file (Java ARchive) is formatted the same as a ZIP file, so you can use a ZIP utility to look at the contents of ij.jar.

You can convert an internal plugin to a user plugin by copying the source to the plugins folder, adding an underscore to the file name and class name, and changing the package statement to an import statement. For example, to change the RoiManager (Analyze/Tools/ROI Manager) to a user plugin:

1. Copy ij/plugin/frame/RoiManager.java to the plugins folder.
2. Change the file name to Roi_Manager.java.
3. Open Roi_Manager.java using *Plugins/Edit* and change all instances (4) of "RoiManager" to "Roi_Manager".
4. Change the first line from "package ij.plugin.frame;" to "import ij.plugin.frame.*;".
5. Run the new plugin using the editor's *File/Compile and Run* command.

There will be a new *Roi Manager* command in the Plugins menu the next time you restart ImageJ.

Changing Location of Plugins Directory

The "plugins.dir" property specifies the location of the parent of the plugins directory. This property can be set from either the command line or from within a Java program that starts ImageJ. For example, if you run ImageJ with the command

```
java -Dplugins.dir=/Applications/ImageJ -cp ../ij.jar:. ij.ImageJ
```

it will look for the plugins folder in /Applications/ImageJ. This property can also be set in a program that launches ImageJ.

System.getProperties().setProperty("plugins.dir",   "/Applications/ImageJ");
new ImageJ(null);

Install Plugin...

Installs a plugin in a user-specified submenu. Plugins with a showAbout() method are also automatically added to the *Help/About Plugins* submenu.

Use the first popup menu to select the plugin and the second to select the submenu it is to installed in. The command must be different from any existing ImageJ command. *Shortcut* (optional) must be a single letter or "F1" through "F12". *Argument* (optional) is the string that will passed to the plugin's run method.

Record...

The easiest way to create a plugin is to open the command recorder, record a series of commands, and then click *Create Plugin*. In record mode, each menu command you use generates a call to ImageJ's run() method. This method has one or two string arguments. The first is the command name. The optional second argument contains dialog box parameters.

Create a rectangular, oval or line selection and the recorder will generate a makeRectangle(), makeOval() or makeLine() method call. Click on "Auto" or "Set" in the *Image/Adjust/Threshold* window to generate a setThresold() call, and on "Reset" to generate a resetThresold() call. Select an image from the Window menu to generate a selectWindow() call. Click in the *Image/Colors* window to generate a setForegroundColor() call and alt-click to generate a setbackgroundColor() call.

New...

Opens a new text window containing a prototype (as Java source code) for one of the three types of plugins supported by ImageJ.

PlugIn: Opens, captures or generates images. Implements the PlugIn interface. The prototype displays "Hello world!" in the ImageJ window. Another example is the Step Maker plugin at http://rsb.info.nih.gov/ij/plugins/steps.html.

PlugInFilter: Processes the active image. Implements the PlugInFilter interface. The prototype inverts the active image twice. Another example is the Image Inverter plugin at http://rsb.info.nih.gov/ij/plugins/inverter.html.

PlugInFrame: Displays a nonimage window containing controls such as buttons and sliders. Extends the PlugInFrame class. The prototype opens a window containing a text area. Another example is the IP Demo plugin at http://rsb.info.nih.gov/ij/plugins/ip-demo.html.

The text window created by this command has two menus: File and Edit. Use *Compile and Run* in the File menu to compile and run the plugin. The Edit menu does not contain *Cut/Copy/Paste* but the keyboard shortcuts for these function can be used. Note that the name you choose for the plugin must include at least one underscore.

Edit...

Opens a text window that allows you to edit, compile and run plugins. Like the the *Compile and Run* command, it requires that ImageJ be running on a Java Virtual Machine that includes the javac compiler.

Compile and Run...

Compiles and runs a plugin. Requires that ImageJ be running on a Java Virtual Machine that includes the javac compiler. Javac is included with the Windows and Linux versions of ImageJ that come bundled with a Java runtime. It is also included with Mac OS X Java. Users of Sun's Java 2 SDK (Software Development Kit) for Windows, Linux and Solaris must add tools.jar to the command line that runs ImageJ. Macintosh users must install Apple's Java SDK.

Here is an example Windows command line for running ImageJ using the Java 2 SDK (aka JDK):

```
java -mx100m -cp ij.jar;C:\jdk1.4\lib\tools.jar ij.ImageJ
```

It assumes the Java 2 SDK is installed in C:\jdk1.4. On a Unix system, the command would look something like this:

```
java -mx100m -cp ij.jar:\usr\local\jdk1.4\lib\tools.jar ij.ImageJ
```

The -mx100 options specifies that ImageJ can use up to 100MB of RAM. To avoid virtual memory thrashing, this value should not be set to more than 2/3 of available RAM (e.g. -mx170m on a 256MB machine).

On Windows, you can create a double-clickable shortcut that uses Java 2 to run ImageJ:

1. Right-click on the desktop and select New->Shortcut from the menu
2. Enter
   `javaw -mx100m -cp ij.jar;C:\jdk1.4\lib\tools.jar ij.ImageJ`
   as the "Command line"; click "Next"
3. Enter a name for the shortcut (e.g. "ImageJ"); click "Finish"
4. Right-click of the newly created shortcut and select Properties from the menu
5. Click on the Shortcut tab
6. Enter the path to the ImageJ folder (normally C:\ImageJ) in "Start in"; click "OK"

"javaw" is a variation of the java command that runs Java applications without a DOS window.