Developing with Adobe Photoshop

Cross-application plug-in development

For many developers, expanding their plug-in to work in other applications across different markets seems cumbersome. Accounting for different code resources and different requirements for multiple applications is a real burden for what often is a small, already taxed research and development department.

It doesn't have to be that way. Currently, a Filter plug-in module written to the Adobe Photoshop 3.0.5 Application Plug-in Interface (API) specification works with five different Adobe applications, expanding your opportunities beyond Adobe Photoshop software's primary "imaging and design" market.

Cross-application plug-in development is a win-win situation. We estimate that with 10-20% more development time for tasks such as adding specific resources and providing alternate code for older host-versions, a plug-in written for the Adobe Photoshop application can be optimized for After Effects, Adobe Illustrator, PageMaker, Adobe PhotoDeluxe and Adobe Premiere. That's an excellent return on your R&D investment; and it's a great win for the customers, who can use your plug-ins in all their graphics applications.

This column begins with our recommended cross-application plug-in development approach, and charts the current status of which API structures are supported by Adobe graphics applications. Future columns will provide more detailed instructions for adapting a plug-in to a specific application. If you just can't wait that long, all this information is available in the Adobe Photoshop Software Development Kit (SDK), and the individual SDKs for other Adobe products.

The different SDKs are available from Adobe's web site: **www.adobe.com** under "Developer Relations".

We recommend you follow this process for your cross-application plug-in development:

Table 1: Process for cross-application plug-in development

Step	Process	Example
1	Assess and determine the problem your plug-in will solve.	We need a Filter plug-in that makes every Nth pixel a specified color.
2	Acquire the primary SDK for your base development.	Download the Photoshop SDK from www.adobe.com.

Table 1: Process for cross-application plug-in development (Continued)

Step	Process	Example
3	Examine the examples and read the API specification in the primary SDK.	Read the initial chapters and the chapter on Filters in the Photoshop SDK. Browse through *PIFilters.h*, *PIGeneral.h* header files; read through example code for *Dissolve* Filter.
4	Determine your development strategy for your base application.	We'll produce a plug-in for both PPC and 680x0 machines, and we need it to work with version 2.5.
5	Read the information in the *Plug-in Resources Guide.pdf*, the Cross-Application Plug-in Development Resources Guiide, with the needs of your plug-in in mind.	Originally, we were only going to parse the filterRect and apply our coloring function. We're interested in dynamic resources, and are considering other features.
6	Reassess your development strategy for your base application.	Our programming goals have changed to include extensibility for changing saturation levels over a period, such as in a QuickTime movie. We will need to modify our initial approach to Photoshop to include variables for saturation, instead of static values.
7	Determine any host requirements for the other target applications.	Because we want to be compatible with version 2.5, we will need to include alternate code that does our job without using the newer callback suites. We also need to make our saturation variable accessible to the dynamic resources that are used by the After Effects and Adobe Premiere applications.
8	Create and program your plug-in.	"NthPixelChange" by MySoftware, Inc.
9	Test under your base application.	Test "NthPixelChange" in the Adobe Photoshop 3.0.5 software.
10	Program and optimize based on your testing results.	We found that we can use the Image Services suite more extensively to help us calculate our changed pixels resulting in increased efficiency.
11	Test under the other target applications.	We added the 'ANIM' and 'FltD' resources and tested in the After Effects and Adobe Premiere software.
12	Modify and optimize based on those results.	Premiere doesn't support the Image Services suite, so we created a check and a branch to implement our original code, if ImageServices is unavailable.
13	Implement your beta-testing program.	A handful of associated development partners were given our plug-in to test.
14	Reassess and modify as needed.	Our beta testers reported bugs and we fixed the errors.

Table 1: Process for cross-application plug-in development (Continued)

Step	Process	Example
15	Package and release your product.	Initial product roll-out.

Table 2 is a list of Adobe applications that support the API outlined in the Adobe Photoshop v3.0.5 SDK. Refer to the *Plug-in Resources Guide.pdf* or the individual product SDKs for specific implementation issues and emulation caveats.

Table 2: Adobe Photoshop API host emulators

Software	Photoshop API version supported	Photoshop modules supported
After Effects	3.0	Filter, Format
Adobe Illustrator	3.0.4 subset; Mac only.	Filter, Format
PageMaker	3.0.4	Filter
Adobe PhotoDeluxe	3.0.4 LE	Acquire, Export, Filter, Format
Adobe Premiere	2.5	Filter

In future columns, we'll discuss specific implementation issues with the different major Adobe applications and offer detailed guidelines for making a plug-in forward- and backward-compatible.

###