IS YOUR SOFTWARE DEVELOPER KIT A SOLUTION OR A PROBLEM?

When implementing a camera into your application, an SDK that is accessible, stable, and well-supported by its manufacturer is as critical as quality hardware.

Martin Missio, *PCO-TECH Inc.*

Software developer kits (SDKs) represent a vital cog in the instrumentation ecosystem, providing developers with the tools, documentation, and guidance to create software applications on a specific platform. Typically, each kit is specific to the manufacturer of that instrument — in this example, a PCO camera.

However, such kits can be a cause for consternation among users who might worry about installation or operation being too difficult, about facing hurdles in operation, or even a change of hardware elements.

This article illustrates how camera SDKs are used and discusses attributes one should seek in both an SDK and the support system implemented by its supplier. It also introduces PCO's approach to SDK fit and functionality.

BROAD FUNCTIONALITY ≠ USER DIFFICULTY

Users can feel overwhelmed, initially, when downloading an SDK to find its accompanying manual/documentation covering hundreds of pages, describing 100+ functions. For the most part, users just want to get started: they'd like some detail on the fundamental concepts first (e.g., opening communication to the camera), then guidance in getting set up and capturing first images.

For this reason, for example, the pco.sdk manual has been structured with key functions up front, along with code examples, allowing users to quickly procure initial images, as well as ensuring a clean exit. As users venture into more specialized camera functions, the manual provides clear instructions and images, reinforcing that everything you could possibly want to do with the camera, you can do using this set of tools.

Still, SDK user difficulties can stem from a number of places, depending on the user's experience and goals. For example, if the user has already integrated cameras in the past and they're switching over to a new SDK, that's typically a reasonably straight-forward transition.

Startup problems generally only arise when someone new to the system is in over their head (e.g., a grad student given this camera and a professor's directive: "Make this thing work with the application we're building," and they don't know how to get started). Once implementation is complete, user queries generally stem from teams trying to get their camera to work in synchronization with some other equipment, where there may be concerns about timing or a novel approach to the task.

BUILDING A USEFUL SDK

PCO supports camera installation through an easily accessible process that begins on the PCO website. Locate your preferred camera model and a link will lead you to the support page where the SDKs, documents, and other support materials are available.

PCO wants people to use these cameras in any way they can envision, so the integration and installation process is made as open, well-documented, and easily navigable as possible. This leads to greater user confidence from the start, necessitating fewer technical support calls — generally speaking, people can just download pco.sdk and start working with it.

Such SDK accessibility is unique. Camera companies have historically been known to charge a fee of up to $1,000 just to have access to their SDK; while companies further complicate the integration endeavor by making you sign an agreement stating that anything you develop with the SDK becomes their property. It goes without saying here that keeping everything related to camera control under lock and key and restricting free flow of information can inhibit progress between a manufacturer and users. Indeed, many PCO customers download the SDK and start looking at it before they've even tested a camera.

If PCO support does have to resolve a customer issue (the customer will fill out a support form or, if it's a familiar customer, they may reach out directly), remote sessions are step one. During these sessions, PCO experts are able to log in (to the client's system) and watch what the client is doing, as well as how the camera(s) is/are reacting.

In some cases, people are using camera functions in a complex way that PCO has never encountered before. They'll come up with a bug and report it to PCO technical support. That support team then works with the pco.sdk developers to resolve the quandary before returning to the customer with a potential fix. It is difficult to overstate the value of working with a support team who understands what is being attempted on the integration side, in addition to knowing the ins and outs of the SDK.

SETUP AND IMPLEMENTATION

Using the pco.sdk to run a camera in your application requires four basic steps: establish communication, complete preliminary setup, take the images, and disconnect the camera.

Certain fundamental elements are common to all cameras, and PCO has structured its interface around that, allowing you to establish a line of communication between your system and a camera — or multiple cameras, even ones not made by PCO. The second step is to request identification from the camera, so to speak: What's its serial number, so it can distinguish it from other cameras that may be connected to the same system?

After that step is complete, you set camera parameters, such as exposure time, to suit your needs. If you want a subset of the full sensor capability, or to designate a region of interest, you will set that up during this step. During this step, you will specify any special synchronization features, such as triggering, as well as the subsequent demands (e.g., you set a command to trigger the capture of an image or a series of images, then transmit the results to the system).

Finally, after setup and usage, it's important to observe proper shutdown procedures to avoid any memory leakages or unwanted system crashes. Typically, the shutdown process is outlined in a structured manner, detailing how to disconnect the camera when you're finished using it and freeing up resources the camera would have been using.

DAILY USE AND IMPLEMENTING HARDWARE UPGRADES

The use of pco.sdk, like other such kits, requires some programming experience. While pco.sdk is written by programmers for programmers, some of the techniques used to make an efficient interface may not be familiar to a novice programmer.

Still, the same SDK is used to bring camera functionality to LabVIEW or MATLAB. Accordingly, if someone knows how to do some rudimentary programming in MATLAB, LabVIEW, or Linux, pco.sdk

probably will not present them with an obtrusive challenge. Again, PCO attempts to provide enough examples and documentation that people can feel comfortable with pco.sdk relatively quickly.

While the SDK is backwards compatible with older PCO camera models, it has evolved over the years to accommodate new camera features and the ability to easily and efficiently stream images using our advanced pco.recorder technology. This plug-and-play utility is made possible by an SDK that's been fundamentally the same for nearly 20 years.

For example, as a real-use case, an individual who'd performed some programming work for PCO in the mid-2000s recently was contracted by a third party to integrate some PCO cameras for a client. She was able to accomplish the task with relative ease even though she'd never seen that particular camera model before. Because of the way that the platform is written, there is commonality between all cameras — new models are designed to fit right into an established, proven, system operation model. In short, pco.sdk is designed to never become obsolete.

This is ideal when people have something that's stable and working, as they typically don't want to download a new version for fear that it might destabilize any elements of the system — which it should never do. Accordingly, pco.sdk has functionality for all its cameras rolled in.

Most new SDK version releases are driven by the introduction of new camera hardware, a new camera interface, or new camera features (e.g., the new camera has some specialized controls for temperature modulation, so those features are added to the SDK). Thus, relevant to the SDK's higher-level language, most functions are transparent and taken care of, but its specialized tools are available if you want to delve into the fine granules of a very specific setup.

CONCLUSION

Hardware can be phenomenal, but if you can't make it work in your application for the lifetime of your system, then it's just a paperweight. A fundamental part of the camera experience should be the accessibility and flexibility to bend each camera to serve the user's application in the most convenient and accurate manner.

The pco.sdk has proven efficient, easy to use, and stable across nearly two decades — a testament to the developers who wrote its original framework, as well as those who currently maintain it, update it, and keep documentation language consistent over time.

More fundamentally, pco.sdk emphasizes organizational commitment to being a valuable partner, rather than just a supplier. PCO-Tech devotes a lot of resources — employing full-time, in-house developers toward an effort that incurs no costs for clients — with this big picture in mind.

To learn more about pco.sdk and the cameras it controls, or to discuss how PCO can help you meet the needs of your application, visit www.pco.de/software/development-tools/pcosdk/ and direct any inquiries to info@pco-tech.com.

ABOUT THE AUTHOR

Martin Missio is director of technical services at PCO-TECH Inc. He has more than 20 years of industry experience and holds bachelor's and master's degrees in science from the University of Waterloo. To speak with Missio about SDKs, contact him at martin.missio@pco-tech.com or 1-866-678-4566 x242.

For more information, visit www.pco-tech.com/intensified-cameras/ and direct any inquiries to info@pco-tech.com.