The FlashPix™ Image File Format

Christopher R. Hauf and J. Scott Houchin
Eastman Kodak Company, Rochester, New York

Abstract

Customers of all types want to use their color images; they want to use the images in compositions, to enhance their communications, to share them easily across the Internet and to make them part of their recorded memories. However, they currently find it too difficult to perform these types of tasks.

The photographic and computer industry is faced with the challenge of delivering compelling digital imaging product solutions to consumers and SOHO computer users that make using color photographic images commonplace on desktop computers. This challenge has gone unmet—until today.

The *FlashPixTM* image file format, developed in collaboration with leading industry partners Hewlett-Packard, Microsoft, and Live Picture Inc., provides the foundation for delivering better digital imaging solutions. It builds on the best features of existing formats (*Kodak Photo CD Image Pac* file format, Live Picture IVUE, TIFF, JPEG, etc.), while adding other powerful new features. With the right software, it can provide an experience where the user does not need to be concerned with (or even understand) resolution or pixels. Meanwhile, it provides the platform for applications to deliver more efficient, faster, and more consistent image-enabled solutions.

The *FlashPix* format truly serves as an open, cross-platform format that will lead to the rapid introduction of new color digital imaging products for consumers and business users. The format also includes higher level functions so applications can use optional viewing transforms to perform simple edits (such as rotation, color correction, cropping, etc.). This capability enables faster , real-time image editing on standard desktop computers and permits the processing of the "transformed image" to be deferred until print-time or performed remotely.

The presentation and paper will describe the technical detail behind this revolutionary concept that allows computer users to easily use color photographic pictures.

The *FlashPix* format is defined in a specification and a test suite developed and published by Kodak in collaboration with Microsoft, Hewlett-Packard, and Live Picture Inc. Only products that meet the specification and pass the test suite may use the *FlashPix* file format name.

Background

Everyone loves images. From the small realtor who wants to put images in a flyer to showcase rental properties to a child who wants to really show where he or she spent their summer vacation, everyone would like to make image-rich documents. For years, the industry has come up with one imaging system after another, all focused on the best way to *store* images. Unfortunately, as we have seen, these systems do not provide a very good way to *use* images.

In early 1995, Kodak and other key industry partners embarked on a program to solve this problem. Led by Eastman Kodak Company, the Microsoft Corporation, the Hewlett-Packard Company, and Live Picture, Inc., joined a cooperative effort and was tasked to design a new image format that would enable the industry to deliver desktop computer solutions that make it easy, enjoyable, and commonplace for people to use digital color photographic images in their offices and homes. The keystone of the effort is the *FlashPix* image file format.

A Day in the Life

The Way it is Now

Today, it can be very difficult and time consuming for consumers and small office/home office (SOHO) users to add digital photographic images to documents. Systems require that the user select the needed resolution upon insertion, thus creating a need for information a user may not have or even understand. Most often, users select the highest resolution available. Placing these large images into documents is often nothing less than a nightmare. If the system can even handle an image that large, the image will take up a large fraction of the system resources causing overall performance to drop dramatically. At that point, the system may even reach the point of unusability. Also, as image acquisition is often totally independent of printing, the typical user will not get the expected color quality out of a printer. With or without user modification of the color balance, the process is usually trial and error. More complex edits such as cropping and arbitrary rotation (for example, to adjust the horizon) can compound the problem since these operations can take several minutes each on a low-end PC.

The Way it Will Be

Tomorrow, the end-user experience will be entirely different. Enabled by the *FlashPix* image file format, the industry creates a whole product solution, consisting of image acquisition methods, image editing applications, image display parts and servers, and image printing systems. Instead of the user being required to determine the needed resolution, image display applications will determine the value automatically, selecting the best resolution for both screen display and printing. Because the image display application chose an appropriate resolution for screen display, placing an image in a document does not take up

significant resources and allows the application to perform normally. Image acquisition is now tightly coupled to image output, so the user will get greatly improved color quality. As the image editing applications are also "resolution smart," operations such as cropping and arbitrary rotation take only a few seconds each; the operation is performed at screen resolution, storing a script of the edits for application on print resolution at print time.

Format Overview

Instead of starting from scratch, the *FlashPix* team drew upon many pre-existing solutions. The *FlashPix* format uses Microsoft's OLE Structured Storage as a container. Image data can be stored in the well known *Kodak Photo-YCC* color interchange space, or a new monitor-ready 8-bit RGB color space, NIFRGB. Image data can be stored uncompressed, or compressed using a JPEG format supplied by Hewlett-Packard Company which is also compliant with the industry standard JPEG format. The *FlashPix* format allows the largest set of pieces of non-image data (such as photographer, scanning application, CCD CFA pattern) to be available of any format. Additionally, the *FlashPix* format adds a layer to specify a viewing transform, allowing a select set of edits to be stored as a script in the file. The format is also designed to facilitate extensions.

Image Data Organization

The *FlashPix* format is a multiresolution image file format. Instead of the image being stored as a single two-dimensional array of pixel values, the image is stored as a series of independent arrays, each representing the image at a different spatial resolution. This allows the image to be displayed on different output devices (monitors, printers, etc.) at different resolutions with minimal resizing of the image.

The *FlashPix* format requires that the separation between consecutive resolutions be 2Ξ in both the horizontal and vertical direction. This both allows a simple decimation scheme for creating the hierarchy and provides a good compromise between performance and file size. In addition, the *FlashPix* format mandates that decimations use a centered alignment method to provide for a known alignment between the original resolution and smaller resolutions. The *FlashPix* format also mandates that when decimating an image with an odd dimension (for example, a 127 pixel wide image), the size of the decimated image is to be rounded up (in this case, the decimated image will be 64 pixels wide).

With an intelligent software layer on top of the format, this multiresolution ability is achieved transparently to the user.

The storage format of each individual resolution can be another obstacle in reading the minimum amount of image data. Traditionally, image data is stored in some form of global row major order in which one entire row of the image is stored in a contiguous block with respect to previous or successive rows. If only a portion of the image is desired, the reader must somehow "skip over" the unused data. If the image data is compressed, this may not be possible without reading and decompressing a large amount of unused data. *FlashPix* file format addresses this problem by tiling the data. Instead of grouping the data into rows, the data is broken into equal size rectangles. Each rectangle represents one tile. The core *FlashPix* specification requires that all tiles be 64 pixels square.

In addition to specifying the organization of image data, the *FlashPix* format also defines a resolution-independent coordinate system for describing the locations of points in the image. The format defines the height of the full-resolution images as 1.0. The width of the image is defined to be the aspect ratio ($R = W/H$). The height of other resolutions in the hierarchy in resolution-independent coordinates are corrected for any rounding of the image size. For example, if the full-resolution image had a height of 127 pixels, the next smaller resolution has a resolution-independent height of $1.0 * 64/63.5$. This correction preserves the alignment between resolutions. By using this coordinate system, an intelligent software layer on top of the file format can further isolate the end user from actual pixel measurements.

Structured Storage

Every day, people find more and more uses for images, often desiring to tie additional information to an image. For example, several digital cameras allow the user to record sound along with the image. It is desired that the *FlashPix* format easily allow additional information to be stored inside the *FlashPix* file itself. It is also important that the format enable tomorrow's desktop systems to easily and automatically catalog *FlashPix* files based on information contained in the file.

A structured storage format was chosen as the "wrapper" for *FlashPix* files. The *FlashPix* format uses Microsoft's OLE Structure Storage format and provides compatibility with other structured storage paradigms such as Bento. Structured storage can be likened to a "file system in a file"; a structured storage file contains both storages (directories) and streams (files). Any application that understands the structured storage format can add additional storages and streams to any structured storage file at will, which will be "invisible" to the original application (unless the application attempts to catalog the file and is written to specifically complain when additional objects are found). A *FlashPix* file consists of a number of storages and streams, which can provide independent access to different parts of the file, such as a file system can provide access to multiple files at once.

Microsoft has also developed a format for storing unordered lists of values, called property sets (property sets are very similar to TIFF tags). Property sets are used for almost all non-image data in a *FlashPix* file. Each property set is described by a GUID (a form of UUIDs defined by the RPC specification.[1] Each set contains an unordered list of values, each of which is described by a four-byte identification code (which is specific to the property set description), a type, and a value. In general property sets, a property may have different types depending on the value

to be stored; however, in a *FlashPix* file, the type of all properties is mandated by the specification.

Microsoft has provided both the specification for the structured storage format[2] and the property set format[3] to the general public. Software libraries for reading and writing libraries are supplied to the public as part of OLE 2.0. In addition, Microsoft has released a reference implementation of the structured storage libraries which can be compiled and used on platforms that are not OLE enabled (for example, Solaris or HP-UX).

Color

The reproduction of color on the desktop has become paramount in recent years as 24-bit video cards and low cost, high-quality color printers and scanners have enabled everyday consumers and small business people to do what used to be reserved for high-end imaging professionals. This trend, however, has highlighted problems in the control of desktop color. The *FlashPix* format attempts to address many of these problems in its design.

The *FlashPix* format provides an extensible color format that includes two unambiguous color space definitions, and optional support for the embedding of International Color Consortium[4] (ICC) color management profiles.

Provided in the *FlashPix*format specification are the definitions for both the *Kodak PhotoYCC* color interchange space and NIFRGB. In combination with the provided colorimetric definitions, *PhotoYCC* space and NIFRGB are further defined in terms of two respective reference viewing environments, which allows for the unambiguous specification of the color appearance of the colors within an image when viewed in their respective viewing environments. The color space definitions in the *FlashPix* format follow the same template used to define the Profile Connection Space (PCS) used in the ICC framework to ensure compatibility and ease of transformation.

The *FlashPix* format provides a strong format for the encoding and storage of the colors within an image. The format is intended to address many of the concerns of today, some of the concerns of tomorrow, and be extensible to handle the concerns of users both today and tomorrow.

Compression

As with almost all existing image formats, the ability to compress image data is important. It is especially important in the *FlashPix* format, as the hierarchy of resolutions combined with the small disks often found on low-end machines can produce an unusable system if the image data is not sufficiently compressed.

The basic compression block in a *FlashPix* file is the tile—it is important that a reader be able to access individual tiles, so any compression must not preclude accessing only a single tile. The *FlashPix* format allows three compression methods: uncompressed, single color compression, and JPEG compression. Each tile is compressed individually from all other tiles, both in that resolution and in all resolutions; each tile may be compressed using a different compression method. This allows a writer to determine the best method (for either size or reconstruction speed) depending on the data in that tile.

The first compression method is uncompressed. Uncompressed tiles are stored in a pixel interleaved, row major organization. This method is useful when storage space is plentiful, the storage device is fast, and the processor speed is slow. If the image must be stored lossless, this is the only method available at this time.

Single color compression is useful when all pixels in a tile are the same color (or can be quantized to the same color). For example, an image with a premultiplied opacity channel may have large areas that are fully transparent. Compressing these tiles using the single color method will provide a 4096:1 compression ratio and a significant performance improvement upon reading the tile.

For tiles that contain real image data and need to be compressed, The *FlashPix* format provides standard JPEG compression, with a few minor enhancements. One problem with off-the-shelf-JPEG solution is the inclusion of quantizer and Huffman table data in a compliant JPEG stream. In a *FlashPix* file, a 2K βψ 3K image would consist of a total of 2048 tiles. The inclusion of the table data in each tile represents significant overhead. To solve this problem, The *FlashPix* format splits the JPEG stream into two parts. The image data is stored in the standard JPEG abbreviated format,[5] which contains only 37 bytes of overhead for each tile. The table data for a particular tile is stored in the standard JPEG abbreviated format for table data,[5] which can be shared between multiple tiles. The *FlashPix* format allows up to 255 abbreviated table streams, which are shared between all resolutions in a single file.

To decompress a JPEG compressed tile, the reader must first load the correct table stream into the codec, and then load the compressed image data. In many cases, the reader will only have to load the table data once, as most images will only use on set of quantizers and Huffman tables for all tiles, which can be remembered by the codec.

The *FlashPix* format JPEG is still compatible with codecs that only understand a complete JPEG stream. Given the JPEG abbreviated table stream and the JPEG abbreviated data stream, a complete JPEG stream can be assembled with only two memory copies.

Non-Image Data

The *FlashPix* format also provides for the storage of a large amount of non-image data. It draws upon many other formats (JFIF/SPIFF, TIFF, TIFF/EP, *Kodak Photo CD Image Pac* file format, Advanced Photo System, etc.) for the types of information that should be stored. It is desired that the *FlashPix* format not be disadvantaged over other formats because of a lack of ability to store a particular piece of non-image data. The *FlashPix* format groups non-image data into the following groups, each of which allows many piece of data to be stored: file source, intellectual property, content description, camera information, per picture camera settings, digital camera characterization, film description, original document scan description, and scan device.

It is also desired that future applications be able to extend the non-image data, just as applications can extend the image data itself. For example, an application may be able to automatically characterize image content may desire to store the image characterization, for access by a search engine.

The *FlashPix* Image View

Nothing causes more frustration than having to wait for the computer to finish performing a simple task. Using today's systems and image file formats, a user can spend several minutes making simple corrections to an image such as slightly adjusting the color balance or performing a 5° rotation to adjust the horizon. By itself, a *FlashPix* image object only hurts the situation, as there is now approximately 33% more image data which must be processed. To address this problem, the *FlashPix* format adds one more layer on top of a plain *FlashPix* image object: the *FlashPix* image view.

The *FlashPix* image view provides a way to store a script of a few basic edits to be performed on a *FlashPix* image object. It also provides a place to cache the result of performing the script. The use of the *FlashPix* image view is illustrated as follows. A user opens a *FlashPix* file in a simple image editing application. The application then reads a screen resolution-sized image from the *FlashPix* file and displays it on the monitor. The user then performs a crop, a 5° rotation, and a slight color adjustment, which are applied to the screen resolution image previously loaded. Happy with the result, the user saves the file. At this point, instead of loading the full-resolution image and reapplying the edits (which will take a long time), the application saves a script of the edits in the image view portion of the *FlashPix* file. Because the application has already performed the edits on the screen resolution image, it can also choose to store it as an additional *FlashPix* image object inside the *FlashPix* file.

The user then places the *FlashPix* file in a word processing document. The application can then either read the script and apply it to the appropriate resolution data, or read the previously cached result, both of which happen very quickly because only a small amount of image data is involved.

The user then prints the document. This causes the application to reload the *FlashPix* file at a resolution appropriate for printing. At this time, sufficient data is probably not available in the cache, so the application must load and reapply the edit script to the image data at an appropriate resolution. Although this operation will take significant time, the delay is acceptable at print time as the user is generally not interactively involved in the printing process.

The *FlashPix* format supports the following operations in the image view: result aspect ratio, rectangle of interest, filtering, spatial orientation, colortwist matrix, and contrast adjustment.

The rectangle of interest operation allows the user to select a portion of the image. Spatial operations such as rotation, scaling and skewing are specified through an affine matrix, which maps the result coordinate system back to the source coordinate system. The result aspect ratio operation specifies actual result rectangle in terms of the result coordinate system.

The filtering operation allows the user to sharpen or blur the image on a resolution-independent basis. The amount of filtering is specified as the change in acutance of the complete imaging system, measured in width of the equivalent Gaussian filter. This allows the user to specify a single number that is valid for that image regardless of the resolution selected by the *FlashPix* reader for display on any particular device. The *FlashPix* reader uses this information, in addition to estimates of the capabilities of the input and output systems to reproduce fine image details. Unfortunately, information on device sharpness is not generally available on the desktop today, even though manufacturers have measured it. However, the system is relatively insensitive to these factors, and the internal default values will generally give good results. Kodak desires to sufficiently enable the industry, so that future printer and scanner drivers would be able to provide their own characterization information.

Modification to the color balance and contrast of the image are specified through the colortwist matrix and contrast adjustment value. The colortwist matrix specifies a four by four matrix, which represents the desired change, as applied data in a normalized form of the *Kodak PhotoYCC* color interchange space. The colortwist matrix can be applied to many color spaces by combining it with matrices to convert the source color space to normalized *PhotoYCC* space values and *PhotoYCC* space values to the destination color space. In many situations, such as when *PhotoYCC* space data is to be converted to a monitor RGB space, the colortwist matrix can be combined with the color conversion process and computed with no extra cost in performance.

Adjustments to contrast are specified by a single value, representing an increase or decrease in contrast. From an equation provided in the *FlashPix* format specification, a look-up table can be generated, which is then applied to the image data.

Conclusion

The goal of the *FlashPix* image format is to solve real user's problems—to make it easy to *use* images, not just to *store* them. Image is data stored in multiple, independent, tiled resolutions to allow an application to get only the needed image data (both in terms of desired resolution and desired spatial region) with minimum unneeded effort. Using structured storage as a wrapper provides unlimited expandability, while still providing compatibility with existing systems. Image data is stored in known, well-defined color spaces, allowing applications to more easily guarantee high-quality reproduction. The *FlashPix* image file format offers flexible compression options, which can be tuned for each individual image to provide the best balance between reconstruction time and file size. The *FlashPix* format also provides an image view layer for every image, allowing low-end users to perform a few simple image edits on a resolution-independent basis, without having

to take the time to apply the edits to high-resolution data during the interactive editing process.

References

1. RPC Specification.
2. Microsoft structured storage format, *FlashPix* Image Format Specification, Appendix A, Eastman Kodak Company, 1996.
3. Microsoft property set format, *FlashPix* Image Format Specification, Appendix A, Eastman Kodak Company, 1996.
4. ICC Profile Format Specification, International Color Consortium, Version 3.2, November 20, 1995.
5. Digital compression and coding of continuous tone still images, Part I, Requirements and guidelines, ISO/IEC JTC1 Draft International Standard, 10918-1, Nov. 1991.