Technical Note TN2057

Fill in the size field before calling ICMGetPixelFormatInfo

CONTENTS

This technote discusses why you should always fill in the size field of the `ICMPixelFormatInfo` structure before calling `ICMGetPixelFormatInfo`.

[Jul 17 2002]

Introduction

The Image Compression Manager routine `ICMGetPixelFormatInfo` takes a `PixelFormat` code and a pointer to a `ICMPixelFormatInfo` structure you provide, and fills in this structure with information about a given pixel format.

```
OSErr ICMGetPixelFormatInfo( OSType PixelFormat,
                             ICMPixelFormatInfoPtr theInfo );
```

- `PixelFormat` - A number or four character code identifying the pixel format.
- `ICMPixelFormatInfoPtr` - A pointer to a ICMPixelFormatInfo structure in which the pixel format information is returned.

It is very important for code calling `ICMGetPixelFormatInfo` to fill in the size field of the `ICMPixelFormatInfo` structure first. Initialize this field with `sizeof(ICMPixelFormatInfo)`. In C, the following syntax does this and also sets the rest of the structure to zero:

```
ICMPixelFormatInfo myICMPixelFormatInfo = {sizeof(ICMPixelFormatInfo), 0};
```

If you call `ICMGetPixelFormatInfo` multiple times, reset the size field to `sizeof(ICMPixelFormatInfo)` before each call.

Back to top

Why should I care?

From time to time, new fields are added to the end of the `ICMPixelFormatInfo` structure -- in QuickTime 4.1, `defaultGammaLevel` was added, and for QuickTime 6.0 `horizontalSubsampling` and `verticalSubsampling` have been added.

The Image Compression Manager is careful to not write more bytes than the size field indicates and on return, the size field will contain the number of valid bytes in the data structure.

By filling in the size field you guarantee that the Image Compression Manager won't write past the end of the structure (and corrupt the stack) if your application and the Image Compression Manager were compiled with different versions of the structure.

```
// QuickTime 6.0
struct ICMPixelFormatInfo {
    long          size;
    unsigned long formatFlags;
    short         bitsPerPixel[14]; /* list each plane's bits per
                                       pixel separately if planar */
                                    /* new field for QuickTime 4.1 */
    Fixed         defaultGammaLevel;
                                    /* new fields for QuickTime 6.0 */
    short         horizontalSubsampling[14]; /* per plane; use
                                                1 if plane is
                                                not subsampled */
    short         verticalSubsampling[14];   /* per plane; use
                                                1 if plane is
                                                not subsampled */
};
typedef struct ICMPixelFormatInfo       ICMPixelFormatInfo;
typedef ICMPixelFormatInfo *            ICMPixelFormatInfoPtr;
```

- `size` - The size of this structure. On entry to ICMGetPixelFormatInfo, this indicates how much memory is available to receive the structure; on return it indicates how much data was filled in. On entry to `ICMSetPixelFormatInfo`, this indicates how much valid data is in the structure. Fields after those labelled as valid should be interpreted as containing zero.
- `formatFlags` - A constant (see below) indicating information about the pixel format.
- `bitsPerPixel` - An array that defines the number of bits for each component of a pixel. The element bitsPerPixel[0] contains the number of bits for the first component, bitsPerPixel[1] the number of bits for the second component, etc. The meaning of this parameter depends on the format flag (see below).
- `defaultGammaLevel` - Defines the default gamma level for newly created GWorlds of this pixel format. Pixel formats for video often have `defaultGammaLevel` set to 2.2 (`kQTCCIR601VideoGammaLevel`). Zero means to use the platform's standard gamma level. This field was introduced in QuickTime 4.1.
- `horizontalSubsampling`, `verticalSubsampling` - For planar pixel formats, these arrays indicate the component subsampling for each component. For example, planar YUV 4:2:0 has one Y sample per pixel (subsampling 1,1), and one U and one V sample per square group of four pixels (subsampling 2,2 and 2,2 respectively). Hence the `horizontalSubsampling` and `verticalSubsampling` fields for `kYUV420PixelFormat` will both contain [1,2,2]. Set unused fields to zero. This information enables QuickTime to allocate `GWorlds` for planar pixel formats and set up planar component headers correctly. This field and functionality was introduced in QuickTime 6.

```
formatFlags Constants
```

- `kICMPixelFormatIsPlanarMask` - This mask constant covers the four least-significant bits (0x0000000F). If these bits of `formatFlags` contain 2 or more, the pixel format is planar and `bitsPerPixel[ ]` represents the bits for each pixel component. Otherwise, the pixel format is chunky (not planar) and `bitsPerPixel[0]` represents the bits per pixel. (Set these bits to zero when defining chunky pixel formats.) Chunky pixel formats pack the different components together. For example, 3 pixels of 32-bit ARGB is represented in memory as ARGBARGBARGB. Planar formats pack the different components separately.
- `kICMPixelFormatIsIndexed` - If the pixel format is indexed (which, by definition, means that there are no individual components) then this flag is set. Generally, color modes of 8 bit per pixel or less are indexed.
- `kICMPixelFormatIsSupportedByQD` - If this flag is set, you can call QuickDraw on `PixMap` structures that store this kind of pixel data. On Macintosh, the classic QD pixel formats have this flag set, but not any of the YUV pixel formats. On Windows, more formats have this flag set, because the Windows implementation of QuickDraw needs to support more pixel formats.
- `kICMPixelFormatIsMonochrome` - If this flag is set, the pixel format is not color. This flag was introduced in QuickTime 6.
- `kICMPixelFormatHasAlphaChannel` - If this flag is set, the pixel format contains an alpha channel. This flag was introduced in QuickTime 6.

Here's an example routine that calls `ICMGetPixelFormatInfo` to determine if a given pixel format is planar.

```
Boolean IsPixelFormatPlanar(OSType inPixelFormat)
{
     OSErr err;
     ICMPixelFormatInfo outInfo = {sizeof(ICMPixelFormatInfo), 0};

     err = ICMGetPixelFormatInfo(inPixelFormat, &outInfo);

     if (noErr != err)
         return false;  // unknown pixel formats are not planar

     if ((outInfo.formatFlags & kICMPixelFormatIsPlanarMask) < 2)
         return false; // zero means pixel format is chunky; one plane is silly

     return true;      // pixel format is planar
}
```

Back to top

References

ICMGetPixelFormatInfo

Pixel Formats

ICMPixelFormatInfo structure

Back to top

Downloadables

Acrobat version of this Note (52K) Download

Back to top