AMERICAN INTERACTIVE MEDIA

TN22 - Using RGB555 Images

ADID: 59.0

Type: Tech Note

Author: Eric Anderson

Created: March 9, 1989

Superceeds: N/A

Pages: 3

Synopsis: A technote to call attention to special handling requirements for RGB555 images. Use of the UCM function dm_write() is discussed along with IFF storage format considerations, and the use of RGB images in real-time files.

A Philips/PolyGram Corporation

11111 Santa Monica Boulevard. Suite 1000. Los Angeles. California 90025
(213) 473-4136 TELEX 182460 POLYPIX LSA FAX (213) 479-5937

AMERICAN INTERACTIVE MEDIA

AIM Technical Note

TN #22: Notes on Using RGB555 files

Written by: Eric Anderson March 9, 1989

This TechNote is to call your attention to special handling requirements for RGB555 images. Use of the UCM function dm_write() is discussed along with CD-I IFF storage format considerations, and the use of RGB images in real-time files. **Note that all references in this technical note are to CD-I IFF specification version .99.**

Image File Format

The CD-IFF specification calls for RGB555 pixels to be split into two banks - the first bank consists of all upper pixel bytes for plane A (PlaneA bytes), the second bank consists of all lower pixel bytes for plane B (PlaneB bytes). The file will look like (AAAA...BBBB...) and is commonly refered to as "planar" format. The specification was written as such to optimize the loading of RGB image data in real-time into the appropriate color plane without unnecessary overhead.

Using dm_write() with RGB555 images

Since there is a difference in storage formats between what dm_write() expects and what is stored on disk in CD-I IFF format, any utility that loads RGB images for display must re-interleave the bytes of the image into chunky format before calling dm_write(). The alternative, of course, is to read the image directly into the drawmap buffers allocated in each plane.

dm_write() usage

dm_write(), as stated in the Green Book [cf VII.2.3.4.1], requires that all bits for a pixel be stored contiguously in the input buffer passed to the routine. For RGB555 images, all 16 bits of a pixel value are placed together. dm_write() splits the pixels into two bytes and writes the upper bytes into plane A drawmap buffer and the lower bytes into plane B drawmap buffer. The buffer will look like (ABABAB....) and is commonly refered to as "chunky" or "interleaved" format.

Image data padding

For rectangular images, the CD-I IFF specification requires that the start of each line of pixels begin on a longword boundary. Since RBG pixels are stored as two banks of byte values, upto 3 padding bytes can be used within a line of the image file. Therefore, the maximum number of padding bytes in the image file would be: (3*imageHeight*2). Note that list oriented images (run-length) are not stored with pad bytes.

Like other images types stored in CD-I IFF format, the specification provides a means for determining the number of pad bytes used. The value *ihdr_line_size* found in the image header chunk ("IHDR") specifies the physical width in bytes of one scan line of image data including required pad bytes to force the lines onto longword boundary. The value *ihdr_width* specifies the logical width of the image in pixels. The number of pad bytes found on each line of image data is therefore (ihdr_line_size - ihdr_width).

RGB555 storage vs. drawmaps

When explicitly allocating storage to contain RGB images, the total size of the storage in bytes is (ihdr_height * ihdr_line_size * 2). dm_create() can be used to create an RGB drawmap by specifying the image height and the phyical width (includes pad bytes). dm_create will allocate two distinct blocks - one each in planeA and planeB of size (height * physicalWidth). Pointers to these two blocks are stored in fields of the drawmap descriptor.

Real Time Record Builder - RTRB

Microware's RTRB makes no assumptions about image formats. The program simply transfers contents of the files to appropriate CDI sector. RGB555 images stored in chunky format would not be displayable in real-time play without spliting the pixels in real-time. RGB555 images stored in planar format can be read directly into drawmap buffers in real-time play with the correct display results.

Real-Time Loading of RGB555 images.

Each of the two banks of the RGB image needs to begin on a sector boundary within an real-time file. PlaneA bytes will be placed on a sector boundary automatically by RTRB. PlaneB bytes must be forced to a sector boundary by RTRB script author with a SEEK command. Seek is used to locate the PlaneB bytes within the file by byte offset from the beginning of the data portion of the image file. The script author is responsible for knowing where the lower bytes are located within the file. Some helpful hints in that regard:

Physical width of image:
(#pad bytes + logical size of image) => physicalWidth
Size of each bank [bytes]:
(physicalWidth * height) => bankSize
Size of RGB image data [bytes]:
(bankSize * 2) => imageSize
Actual number of sectors to hold each bank
(bankSize / FORM2_SECTOR_LENGTH) => sectorCount
Whole sectors needed to contain each bank:
ceil(sectorCount) => wholeScts
note that ceiling function finds smallest integer not less than input.

Instructions to RTRB for an RGB file would look like:

OUTPUT planeA bytes as sequence of whole sectors [= wholeScts]
SEEK to beginning of planeB bytes [offset = bankSize]
OUTPUT planeB bytes as sequence of whole sectors [= wholeScts]