radarhere When reading past the end of a scan line, reduce bytes left History

2 contributors  

129 lines (106 sloc) | 3.06 KB ...

```c
/*
 * The Python Imaging Library.
 * $Id$
 *
 * decoder for Targa RLE data.
 *
 * history:
 * 97-01-04 fl created
 * 98-09-11 fl don't one byte per pixel; take orientation into account
 *
 * Copyright (c) Fredrik Lundh 1997.
 * Copyright (c) Secret Labs AB 1997-98.
 *
 * See the README file for information on usage and redistribution.
 */

#include "Imaging.h"

int
ImagingTgaRleDecode(Imaging im, ImagingCodecState state, UINT8 *buf, Py_ssize_t bytes) {
    int n, depth;
    UINT8 *ptr;
    int extra_bytes = 0;

    ptr = buf;

    if (state->state == 0) {
        /* check image orientation */
        if (state->ystep < 0) {
```

```
            state->y = state->ysize - 1;
            state->ystep = -1;
        } else {
            state->ystep = 1;
        }

        state->state = 1;
    }

    depth = state->count;

    for (;;) {
        if (bytes < 1) {
            return ptr - buf;
        }

        n = depth * ((ptr[0] & 0x7f) + 1);
        if (ptr[0] & 0x80) {
            /* Run (1 + pixelsize bytes) */
            if (bytes < 1 + depth) {
                break;
            }

            if (state->x + n > state->bytes) {
                state->errcode = IMAGING_CODEC_OVERRUN;
                return -1;
            }

            if (depth == 1) {
                memset(state->buffer + state->x, ptr[1], n);
            } else {
                int i;
                for (i = 0; i < n; i += depth) {
                    memcpy(state->buffer + state->x + i, ptr + 1, depth);
                }
            }

            ptr += 1 + depth;
            bytes -= 1 + depth;
        } else {
            /* Literal (1+n+1 bytes block) */
            if (bytes < 1 + n) {
                break;
            }

            if (state->x + n > state->bytes) {
                extra_bytes = n; /* full value */
                n = state->bytes - state->x;
                extra_bytes -= n;
```

```
            }

            memcpy(state->buffer + state->x, ptr + 1, n);

            ptr += 1 + n;
            bytes -= 1 + n;
        }

        for (;;) {
            state->x += n;

            if (state->x >= state->bytes) {
                /* Got a full line, unpack it */
                state->shuffle(
                    (UINT8 *)im->image[state->y + state->yoff] +
                        state->xoff * im->pixelsize,
                    state->buffer,
                    state->xsize);

                state->x = 0;

                state->y += state->ystep;

                if (state->y < 0 || state->y >= state->ysize) {
                    /* End of file (errcode = 0) */
                    return -1;
                }
            }

            if (extra_bytes == 0) {
                break;
            }

            if (state->x > 0) {
                break;  // assert
            }

            if (extra_bytes >= state->bytes) {
                n = state->bytes;
            } else {
                n = extra_bytes;
            }
            memcpy(state->buffer + state->x, ptr, n);
            ptr += n;
            bytes -= n;
            extra_bytes -= n;
        }
    }

```

```
128     return ptr - buf;
129 }
```