

main ▾

...

[Pillow](#) / [src](#) / [libImaging](#) / TgaRleDecode.c

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

[History](#)

2 contributors



129 lines (106 sloc) | 3.06 KB

...

```
1  /*
2   * The Python Imaging Library.
3   * $Id$
4   *
5   * decoder for Targa RLE data.
6   *
7   * history:
8   * 97-01-04 fl created
9   * 98-09-11 fl don't one byte per pixel; take orientation into account
10  *
11  * Copyright (c) Fredrik Lundh 1997.
12  * Copyright (c) Secret Labs AB 1997-98.
13  *
14  * See the README file for information on usage and redistribution.
15  */
16
17  #include "Imaging.h"
18
19  int
20  ImagingTgaRleDecode(Imaging im, ImagingCodecState state, UINT8 *buf, Py_ssize_t bytes) {
21      int n, depth;
22      UINT8 *ptr;
23      int extra_bytes = 0;
24
25      ptr = buf;
26
27      if (state->state == 0) {
28          /* check image orientation */
29          if (state->ystep < 0) {
```

```

30         state->y = state->ysize - 1;
31         state->ystep = -1;
32     } else {
33         state->ystep = 1;
34     }
35
36     state->state = 1;
37 }
38
39 depth = state->count;
40
41 for (;;) {
42     if (bytes < 1) {
43         return ptr - buf;
44     }
45
46     n = depth * ((ptr[0] & 0x7f) + 1);
47     if (ptr[0] & 0x80) {
48         /* Run (1 + pixelsize bytes) */
49         if (bytes < 1 + depth) {
50             break;
51         }
52
53         if (state->x + n > state->bytes) {
54             state->errcode = IMAGING_CODEC_OVERRUN;
55             return -1;
56         }
57
58         if (depth == 1) {
59             memset(state->buffer + state->x, ptr[1], n);
60         } else {
61             int i;
62             for (i = 0; i < n; i += depth) {
63                 memcpy(state->buffer + state->x + i, ptr + 1, depth);
64             }
65         }
66
67         ptr += 1 + depth;
68         bytes -= 1 + depth;
69     } else {
70         /* Literal (1+n+1 bytes block) */
71         if (bytes < 1 + n) {
72             break;
73         }
74
75         if (state->x + n > state->bytes) {
76             extra_bytes = n; /* full value */
77             n = state->bytes - state->x;
78             extra_bytes -= n;

```

```

79         }
80
81         memcpy(state->buffer + state->x, ptr + 1, n);
82
83         ptr += 1 + n;
84         bytes -= 1 + n;
85     }
86
87     for (;;) {
88         state->x += n;
89
90         if (state->x >= state->bytes) {
91             /* Got a full line, unpack it */
92             state->shuffle(
93                 (UINT8 *)im->image[state->y + state->yoff] +
94                 state->xoff * im->pixelsize,
95                 state->buffer,
96                 state->xsize);
97
98             state->x = 0;
99
100            state->y += state->ystep;
101
102            if (state->y < 0 || state->y >= state->ysize) {
103                /* End of file (errcode = 0) */
104                return -1;
105            }
106        }
107
108        if (extra_bytes == 0) {
109            break;
110        }
111
112        if (state->x > 0) {
113            break; // assert
114        }
115
116        if (extra_bytes >= state->bytes) {
117            n = state->bytes;
118        } else {
119            n = extra_bytes;
120        }
121        memcpy(state->buffer + state->x, ptr, n);
122        ptr += n;
123        bytes -= n;
124        extra_bytes -= n;
125    }
126 }
127

```

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