PNG-GROUP-DRAFT
PNG-SciVis
Expires: 10 Oct 1996

PNG Development Group
png-list@dworkin.wustl.edu
06 Apr 1996

PNG Scientific Visualization Chunks, draft 0.960406

File

png-scivis-chunks-0.960406.txt

Status of this Memo

This document is an informal draft of the PNG development group.

Comments on this document can be sent to the PNG specification maintainers at `png-info@uunet.uu.net` or at `png-list@dworkin.wustl.edu`.

Distribution of this memo is unlimited.

At present, the latest version of this document is available on the World Wide Web from `ftp://swrinde.nde.swri.edu/pub/png-group/documents/`.

Notices

Copyright © 1996 Thomas Boutell

Permission is granted to copy and distribute this document for any purpose and without charge, provided that the copyright notice and this notice are preserved, and that any substantive changes or deletions from the original are clearly marked.

Abstract

This document describes some special-purpose chunk types that have been proposed for use in various PNG (Portable Network Graphics) applications. They have not yet been approved for registration by the PNG developers, and therefore are experimental and their format is subject to change. The proposed chunks are `drNG/DrNG`, `faLS`, `loGE/LoGE`, `xySC`, and `zsCL`.

Contents

1 Introduction 2
2 Using Proposed PNG Chunks 2
3 Summary of Proposed Sci-Vis Chunks 2
4 Proposed Chunk Specifications 2
4.1 `drNG/DrNG` Pixel display range 3
4.2 `faLS` False-color Palette 4
4.3 `loGE/LoGE` Logarithmic Encoding 5
4.4 `xySC` X and Y-Scale of Image Subject 7
4.5 `zsCL` Z-Scale of Pixel Values 9
5 Security considerations 10
6 Credits 10

1 Introduction

Chunks described here have been proposed to the PNG development group for use by the scientific visualization community. Their definitions are subject to change until such time as the group formally registers them. No decoder is required or expected to implement these chunks except for experimental or evaluation purposes. Comments on these proposals, and new proposals for additional chunk types, should be sent to the PNG specification maintainers at `png-info@uunet.uu.net`. The basic PNG specification is available from the W3C archive at `http://www.w3.org/pub/WWW/TR/WD-png`.

2 Using Proposed PNG Chunks

No chunks described in this document have yet been registered by the PNG maintainers. Therefore they have a lower-case letter in the second position of the chunkname. They are **experimental** chunks and the format is subject to change. If and when any become registered, the second letter of the chunk name will become uppercase and there will be no further change to the format.

Whenever you use any of these unregistered chunks you should also include a `tEXt` chunk describing it, for example:

```
tEXtComment\0
This file contains a zsCL chunk written according to the
format given in Version 0.960406 of the PNG Proposed Chunks document.
```

3 Summary of Proposed Sci-Vis Chunks

This table summarizes some properties of the proposed chunks described in this document.

Chunk name	Multiple OK?	Ordering constraints
drNG/DrNG	No	Before IDAT
faLS	No	Before IDAT
loGE/LoGE	No	Before IDAT
xySC	No	Before IDAT
zsCL	No	Before IDAT

4 Proposed Chunk Specifications

This section provides the detailed specifications for the proposed chunks.

4.1 drNG/DrNG Pixel display range

This chunk may be used when the pixel values do not occupy the full range of possible values, and when bit depth scaling is not appropriate. It can also be used to enhance contrast by scaling to a larger range than the actual range of pixel values. It can be used to correct the brightness of an image that has been scaled by zero-filling rather than linear scaling. See [link to sBIT in core spec]. The chunk defines the pixel values that correspond to minimum and maximum intensity, to which the pixel data is to be scaled.

The chunk can be used as a critical chunk, named `DrNG`, or as an ancillary chunk, named `drNG`. The syntax and function is exactly the same whichever name is used. The critical version should only be used if the image cannot be meaningfully displayed without performing `DrNG` scaling. Decoders not recognizing `DrNG` will not attempt to display the image at all. Encoders are strongly encouraged to scale the image data properly and to use the noncritical version, `drNG`, if at all possible.

This chunk's contents are

min_sample	m bytes	Raw sample value corresponding to minimum intensity of the graylevel or red channel, written as a text floating-point value [link to Floating-Point Values in extensions document]
0	1 byte	a zero byte to separate fields
max_sample	n bytes	Raw sample value corresponding to maximum intensity for the graylevel or red channel

```
(The following entries may be omitted for grayscale images, or for
color images where identical values are to be applied to all three color
channels)

0          1 byte     a zero byte to separate fields

min_green  m bytes    raw sample value corresponding to
                      minimum green intensity

0          1 byte     a zero byte to separate fields

max_green  m bytes    raw sample value corresponding to
                      maximum green intensity

0          1 byte     a zero byte to separate fields

min_blue   m bytes    raw sample value corresponding to
                      minimum blue intensity

0          1 byte     a zero byte to separate fields

max_blue   m bytes    raw sample value corresponding to
                      maximum blue intensity
```

If this chunk is present, graylevel or color sample values are to be scaled for display between minimum and maximum intensity by linear interpolation. When the sample value falls outside the range min_value..max_value, it is to be set to min_value or max_value as appropriate. For each graylevel or color sample, the conversion is

```
ratio := (2^bit_depth - 1)/(max_value - min_value)
result := (input_sample - min_value) * ratio
output_sample := LIMIT (0, result, 2^output_bit_depth-1)

in which the function LIMIT (low, x, high) is
MAX (MIN (x, high), low).
```

Note that min_value and max_value are permitted to be negative, positive, or zero. The only restriction is that they be different from each other.

Encoders **should not** use `drNG/DrNG` in lieu of reasonably scaling the samples. For example, if the sample values range from 0 (black) to 1000 (white), it would be an extremely bad idea to use the `drNG` chunk to request the decoder to scale 1000 up to 65535, because a PNG viewer that does not understand `drNG/DrNG` (and, since this is an extension chunk, most viewers will probably not understand `drNG/DrNG`) would simply display a very dark rectangle. Instead, multiply your samples by 64, and use `drNG` with max_sample=64000 to request the decoder to do the final adjustment.

If the tRNS chunk is present, its value is compared to the unscaled pixel value, prior to applying the drNG/DrNG scaling. The faLS, gAMA, cHRM, and alpha conversions, if present, are applied to the scaled sample values.

If present, this chunk must appear before the first IDAT chunk. Only one drNG/DrNG chunk is permitted in a PNG file.

4.2 faLS False-color Palette

This chunk is used with grayscale data, where the pixels are to be rendered in a false color according to the grayscale value. The grayscale sample of each pixel serves as a pointer into the false-color palette.

This chunk contains from 1 to 2^bitdepth 8-byte false-color palette entries:

```
index:          2 bytes, range 0 .. (2^bitdepth) -1)
red:            2 bytes, range 0 .. 65535
green:          2 bytes, range 0 .. 65535
blue:           2 bytes, range 0 .. 65535

etc.
```

The number of entries is determined from the chunk length. This length not divisible by 8 is an error.

This chunk may appear for color type 0 or color type 4. If it appears for any other color type, it will be ignored.

The complete 2^bitdepth-entry false-color palette can be built from the chunk data. If the first entry (index value 0) is missing, it will be assumed to be {0, 0, 0} (black). If the last entry (index value 2^bitdepth - 1) is missing, it will be assumed to be {65535, 65535, 65535} (white). The red, green, blue samples for other missing entries are filled in by linearly interpolating between the samples that are present, independently for each of the three color components.

Once the complete false-color palette is established, it is used similarly to PLTE. The first entry in the completed false-color palette is referenced by the grayscale value 0, the second by grayscale value 1, etc.

If the tRNS chunk is present, its value is compared to the graylevel, not to the converted false-color of the pixels. If the bKGD chunk is present, background pixels will be displayed in the false-color corresponding to the grayscale value found in the bKGD chunk. The gAMA, cHRM, and alpha conversions, if present, are applied to the color samples in the converted false-color pixels.

Note that the gAMA and other values must be selected so that the grayscale image is correctly displayed when the faLS chunk is unrecognized or ignored.

If this chunk does appear, it must precede the first IDAT chunk. There may be only one faLS chunk in a PNG file.

4.3 loGE/LoGE Logarithmic Encoding

The loGE/LoGE chunk is used to decode image data whose grayscale or color samples have been logarithmically encoded.

The chunk can be used as a critical chunk, named LoGE, or as an ancillary chunk, named loGE. The syntax and function is exactly the same whichever name is used. The critical version should only be used if the image cannot be meaningfully displayed without performing LoGE scaling. Decoders not recognizing LoGE will not attempt to display the image at all. Encoders are strongly encouraged to include a gAMA chunk that permits a meaningful if not completely accurate display, and use the noncritical version, loGE, if at all possible.

The loGE/LoGE chunk's contents are

```
P0              n0 bytes    (First parameter, a real number
                            written as a text floating-point
                            value [link to Floating-Point
                            Values in extensions document]
Null separator: 1 byte

P1              n1 bytes    (Second parameter)

Null separator: 1 byte

P2              n2 bytes    (Third parameter)
```

There is no trailing zero for the final string.

The scaling algorithm is

```
normalized_sample_value := input_sample/(2^input_bitdepth-1)
scaled_value := P0 + P1 * P2^normalized_sample_value
output_sample := LIMIT (0, scaled_value, 2^output_bitdepth-1)

in which the function LIMIT (low, x, high) is
MAX (MIN (x, high), low).
```

For color_types 0 and 4, the normalized sample value is the grayscale value of the pixel normalized to the range [0.0:1.0] by dividing it by the maximum value for the input bit depth, using floating point arithmetic.

For color_types 2, 3, and 6, the scaling algorithm is applied independently to each of the color samples similarly normalized to [0.0:1.0].

The alpha channel, if present, is not affected by the loGE/LoGE transformation. Alpha compositing is done in the normal manner, with the transformed pixels forming the foreground image [link to Decoders: Alpha Processing in core spec]

Pure logarithmic data can be expressed with

```
P0 = 0
P1 = min_val
P2 = max_val/min_val
```

In which the range [min_val..max_val] includes the minimum and maximum values appearing in the source data.

When the image is decoded using the loGE/LoGE data, the gamma calculation for the decoded samples should be done as though the file_gamma were 1.0, regardless of the contents of the gAMA chunk.

It is advisable to include a gAMA chunk with logarithmic data, in case a viewer does not use the loGE data to decode it. A value of gamma should be chosen that allows the image to be displayed as well as it can be with a viewer that supports gAMA but not loGE.

You can select a value of gamma (also called file_gamma) by eye as described in [link to Gamma Tutorial: General Gamma Handling in core spec], or you can calculate one as follows:

If the maximum value of the logarithmically decoded pixels can reasonably be displayed as white on the monitor, then specifying a value of gamma given by

```
gamma = LN( LN(0.2) / LN(max_val/min_val) + 1) / LN(0.2)
(in which LN() is the natural logarithm function)
```

causes the two values max_val and 0.2*max_val to be reproduced at correct brightness on screen - these can be thought of as white and mid-grey. Tones between white and mid-grey will be a bit too bright, ones darker than mid-grey will be increasingly too dark, but it's a reasonable approximation for viewable display. A max_val/min_val ratio of 64 gives a gamma of about 0.3, a max_val/min_val of 1000 gives a gamma of about 0.165, so a viewer assuming a default gamma of 0.45 is going to give a bright, washed-out image for any log-encoded image.

In cases where the maximum sample value is many times brighter than scene white (i.e. the image is encoded to retain specular highlight information), the formula above doesn't apply, but there's probably no way of getting such an image to display reasonably anyway with a viewer that doesn't understand loGE.

If present, the loGE/LoGE chunk must appear before the first IDAT chunk. Only one instance of the loGE/LoGE chunk is permitted in a PNG datastream.

4.4 xySC X and Y-Scale of Image Subject

This chunk relates the actual dimension of the image subject in the "x" (width) and "y" (height) direction to the column and row position of a pixel. It provides some additional capability beyond that available in the sCAL [link to sCAL in PNG extensions document] chunk, which does not provide for an offset, and does not provide for different units in the width and height directions. The latter could be useful, for example, in racing photo-finish "streak" images, where one of the dimensions is time and the other is length. The chunk does not affect the image display, but could be used to construct scaled axes adjacent to the image.

The xySC chunk's contents are

```
Xunit          xu bytes  (Latin-1 symbol or description of
                         the width unit, eg. milliseconds,
                         degrees West Longitude,  etc).
                         A zero-length string may be used if
                         the data is dimensionless.

Null separator: 1 byte

Xoffset        xo bytes  X offset, a real number written as a
                         text floating-point value [link to
                         Floating-Point Values in extensions
                         document] This is the physical value
                         of "x" corresponding to the left
                         edge of the image.

Null separator: 1 byte

Xscale         xs bytes  X scale, the x-distance corresponding
                         to the width of a pixel.  Must be
                         non-zero.

Null separator: 1 byte

Yunit          yu bytes  Latin-1 symbol or description of the
                         height unit, eg. meters, degrees North
                         Latitude,  etc).  A zero-length string
                         may be used if the data is
                         dimensionless.

Null separator: 1 byte

Yoffset        yo bytes  Y offset, a real number written as a
                         text floating-point value.  This is
                         the physical value of "y"
                         corresponding to the top edge of the
                         image.
```

```
Null separator: 1 byte

Yscale          ys bytes  Y scale, the y-distance corresponding
                          to the height of a pixel.  Must be
                          non-zero.
```

There is no trailing zero for the final string.

The scaling algorithm for finding the physical x-value and y-value of the middle of a pixel is

```
physical_x_value := Xoffset + Xscale * (column_number + 0.5)
physical_y_value := Yoffset + Yscale * (row_number + 0.5)
```

If present, the `xySC` chunk must appear before the first `IDAT` chunk. Only one instance of the `xySC` chunk is permitted in a PNG stream.

4.5 `zsCL` Z-Scale of Pixel Values

When a PNG file is being used to store physical data other than color values, such as a two-dimensional temperature field, the `zsCL` chunk can be used to record the relationship between pixel values and actual physical values. The `zsCL` data might be used to construct a reference color bar beside the image. It is not expected to affect the way the image is displayed.

The `zsCL` chunk's contents are

```
Equation type   1 byte   (0 for linear scaling,
                          1 for base-e exponential scaling,
                          2 for arbitrary-base exponential
                            scaling)

N               1 byte   (Number of parameters)

Unit            u bytes  (Latin-1 symbol or description of
                          the unit, eg. K, Population Density,
                          MPa, etc).  A zero-length string may
                          be used if the data is dimensionless.

Null separator: 1 byte

P0              p0 bytes  (First parameter, a real number
                          written as a text floating-point
                          value [link to Floating-Point Values
                          in extensions document]

Null separator: 1 byte
```

```
P1             p1 bytes  (Second parameter)

Null separator: 1 byte   (only if P2 appears)

P2             p2 bytes  (Third parameter, optional)
```

There is no trailing zero for the final string.

The scaling algorithm is

```
if equation_type == 0 then
scaled_value := P0 + P1 * normalized_sample_value

else if equation_type == 1 then
scaled_value := P0 + P1 * EXP(P2 * normalized_sample_value)

else if equation_type == 2 then
scaled_value := P0 + P1 * P2^normalized_sample_value
```

in which EXP(x) is the exponential function, e^x, in which "e" is the base of natural logarithms (approximately 2.718282). For color_types 0 and 4, the normalized sample value is the grayscale sample value of the pixel normalized to the range [0.0:1.0] by dividing it by the maximum value for the bit depth, using floating point arithmetic.

For color_types 2, 3, and 6, the scaling algorithm is applied independently to each of the color sample values similarly normalized to [0.0:1.0].

Pure logarithmic data can be expressed either with

```
Equation_type = 1                  Equation_type = 2
N  = 3                 or with     N  = 3
P0 = 0                             P0 = 0
P1 = min                           P1 = min
P2 = LOGe(max/min)                 P2 = max/min
```

The `zsCL` data is not intended to be used by a decoder to affect the way the pixels are displayed. `zsCL` is simply a comment, and could be used, for example, to construct a reference color bar scale beside the image, or to extract the original physical values recorded in the file.. The `drNG/DrNG` or `loGE/LoGE` chunk should be used if the encoder wants the decoder to modify the sample values for display purposes.

If present, the `zsCL` chunk must appear before the first `IDAT` chunk. Only one instance of the `zsCL` chunk is permitted in a PNG stream.

5 Security considerations

Security considerations are addressed in the basic PNG specification, `http://www.w3.org/pub/WWW/TR/WD-png`.

The same precautions taken when displaying `tEXt` data should be taken when displaying the text contained in the "unit" strings of the `xySC` and `zsCL` chunks. Viewers should not display these strings directly without first checking for the presence of nonprintable characters, and for the <ESC> character in particular.

No known additional security hazards are posed by the chunks described here.

6 Credits

Contributors

Contributors' names are presented in alphabetical order:

- Mark Adler, madler@alumni.caltech.edu
- Thomas Boutell, boutell@boutell.com
- Christian Brunschen, cb@df.lth.se
- Adam M. Costello, amc@cs.wustl.edu
- Lee Daniel Crocker, lee@piclab.com
- Andreas Dilger, adilger@enel.ucalgary.ca
- Oliver Fromme, fromme@rz.tu-clausthal.de
- Todd French, todd@lfd.physics.uiuc.edu
- Jean-loup Gailly, gzip@prep.ai.mit.edu
- Chris Herborth, chrish@qnx.com
- Alex Jakulin, alex@hermes.si
- Neal Kettler, kettler@cs.colostate.edu
- Jim King, jimk@mathtype.com
- Tom Lane, tgl@sss.pgh.pa.us
- Alexander Lehmann, alex@hal.rhein-main.de
- Chris Lilley, chris.lilley@mcc.ac.uk
- Dave Martindale, davem@cs.ubc.ca

- Owen Mortensen, ojm@csi.compuserve.com
- Robert P. Poole, lionboy@primenet.com
- Glenn Randers-Pehrson, glennrp@arl.mil or randeg@alumni.rpi.edu
- Greg Roelofs, newt@uchicago.edu
- Willem van Schaik, gwillem@ntuvax.ntu.ac.sg
- Guy Schalnat, schalnat@group42.com
- Paul Schmidt, pschmidt@photodex.com
- Tim Wegner, 71320.675@compuserve.com

Editor

- Glenn Randers-Pehrson, glennrp@arl.mil or randeg@alumni.rpi.edu

End of PNG Sci-Vis Chunks Listing