MNG-VLC (Multiple-image Network Graphics—Very Low Complexity) Format Version 0.95a

For list of authors, see Credits (Chapter 13).

Status of this Memo

This document is a specification by the PNG development group.

It has been approved by a vote of the group, and future technical changes will require formal approval by a vote of the group. It is the intent of the group to maintain backward compatibility if possible. We will, however, correct any technical deficiencies discovered in the course of developing "beta" implementations.

Comments on this document can be sent to the PNG specification maintainers at one of the following addresses:

- mpng-list@ccrc.wustl.edu
- png-group@w3.org
- png-info@uunet.uu.net

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/mng/documents/.

In the case of any discrepancy between this extract and the full MNG specification, the full MNG specification shall take precedence.

Changes from sixty-third MNG draft (mng-0.95-19990425)

- Started a Revision History section.

Abstract

This document presents the MNG-VLC (Multiple-image Network Graphics, Very Low Complexity) format, which is a proper subset of the MNG (Multiple-image Network Graphics) format.

MNG is a multiple-image member of the PNG (Portable Network Graphics) format family that can contain animations (slide shows) or complex still frames, comprised of multiple PNG single-image datastreams.

The MNG-VLC format uses the same chunk structure that is defined in the PNG specification and shares other features of the PNG format. Any valid PNG datastream is also a valid MNG-VLC datastream.

A MNG-VLC frame normally contains a two-dimensional image or a two-dimensional layout of smaller images. It could also contain three-dimensional “voxel” data arranged as a series of two-dimensional planes (or tomographic slices), each plane being represented by a PNG datastream.

This document includes examples that demonstrate various capabilities of MNG-VLC.

Table of Contents

1 Introduction

This document presents a very-low-complexity version (MNG-VLC, which is a proper subset) of the MNG (Multiple-image Network Graphics) format.

Note: This specification depends on the PNG (Portable Network Graphics) [PNG] and, for MNG-VLC applications that are enhanced with JNG support, the JNG (JPEG Network Graphics) specifications. The PNG specification is available at the PNG web site,

`http://www.cdrom.com/pub/png/`

and the JNG (JPEG Network Graphics) specification and the full MNG specification are available at the MNG web site,

`http://www.cdrom.com/pub/mng/`

A MNG-VLC datastream describes a sequence of zero or more single frames, each of which is composed of a single embedded image, or one single frame that can be composed of zero or more embedded images.

The embedded images can be PNG or JNG datastreams. MNG-VLC datastreams do not contain JNG (JPEG Network Graphics) datastreams, which are allowed in full MNG datastreams, but MNG-VLC applications can be enhanced to recognize and process those as well.

A typical MNG-VLC datastream consists of:

- The 8-byte MNG signature.
- The MHDR chunk.
- Layer definitions. A layer is:
  - An embedded image.
  - The background.
- The MEND chunk.

MNG is fundamentally declarative; it describes the elements that go into an individual frame. It is up to the decoder to work out an efficient way of making the screen match the desired composition whenever a nonzero interframe delay occurs. Simple decoders can handle it as if it were procedural, compositing the images into the frame buffer in the order that they appear, but efficient decoders might do something different, as long as the final appearance of the frame is the same.

MNG is pronounced “Ming.”

When a MNG datastream is stored in a file, it is recommended that “.mng” be used as the file suffix. In network applications, the Media Type “video/x-mng” can be used. Registration of the media type “video/mng” might be pursued at some future date.

The first eight bytes of a MNG datastream are

```
138 77 78 71 13 10 26 10
```

(decimal) which is similar to the PNG signature with "\212 M N G" instead of "\211 P N G" in bytes 0–3.

Chunk structure (length, name, data, CRC) and the chunk-naming system are identical to those defined in the PNG specification. As in PNG, all integers that require more than one byte must be in network byte order.

The chunk copying rules for MNG employ the same mechanism as PNG, but with rules that are explained more fully in the full MNG specification.

Note that decoders are not required to follow any decoding models described in this specification nor to follow the instructions in this specification, as long as they produce results identical to those that could be produced by a decoder that did use this model and did follow the instructions.

Each chunk of the MNG datastream or of any embedded object is an independent entity, i.e., no chunk is ever enclosed in the data segment of another chunk.

An independent PNG datastream, with a PNG signature, is also a valid MNG-VLC datastream that must be recognized and decoded by MNG-VLC decoders. An independent JNG datastream must also be recognized and decoded by any MNG-VLC decoder that has been enhanced to include JNG support. This kind of MNG-VLC or enhanced MNG-VLC datastream will contain only a single embedded image.

Because the embedded objects making up a MNG are normally in PNG format, MNG shares the good features of PNG:

- It is unencumbered by patents.
- It is streamable.
- It has excellent, lossless compression.
- It stores up to four channels (red, green, blue, alpha), with up to 16 bits per channel.
- It provides transparency and an alpha channel.
- It provides platform-independent rendition of colors by inclusion of gamma and chromaticity information.
- It provides early detection of common file transmission errors and robust detection of file corruption.
- Single-image GIF files can be losslessly converted to PNG.
- It is complementary to JPEG and does not attempt to replace JPEG for lossy storage of images (however, JNG-enhanced MNG-VLC can accommodate JPEG-encoded images that are encoded in the PNG-like JNG format.

In addition:

- It provides animation with a constant framing rate.
- It allows composition of frames containing multiple images.

- Multiple-image GIF files (except for those using a variable framing rate, the "restore-to-background" or the "restore-to-previous" disposal method) can be losslessly converted to MNG-VLC.

2 Terminology

See also the glossary in the PNG specification and the "terminology" section of the full MNG specification.

requirement levels
The words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119, and the word "CAN" is equivalent to the word "MAY" as described therein. "NOT ALLOWED" and "NOT PERMITTED" describe conditions that "MUST NOT" occur. "ALLOWED" and "PERMITTED" describe conditions that "CAN" occur.

embedded image
An image that appears in-line in a MNG datastream.

frame
A layout of zero or more layers, with zero interframe delay time between them, followed by a specified nonzero delay time or by the MEND chunk, that is to be displayed as a still picture or as part of an animation. An animation would ideally appear to a perfect observer (with an inhumanly fast visual system) as a sequence of still pictures. A group of MNG images, with zero interframe delay time between them, and also followed by a zero interframe delay time, makes up a "subframe" (see below) instead of a "frame". The final subframe in the datastream completes a frame, regardless of its interframe delay. In MNG-VLC datastreams, each layer (except for the background layer and the first image, which are combined into one frame) is a frame, unless the framing rate (from the MHDR ticks-per-second field) is zero. When the framing rate is zero, the entire datastream describes a single frame.

When the layers of a frame do not cover the entire area defined by the width and height fields from the MHDR chunk, the layers are composited over the previous frame to obtain the new frame.

frame origin
The upper left corner of the output device (frame buffer, screen, window, page, etc.) where the pixels are to be displayed. In MNG-VLC, all images must be placed with the image's upper left corner at the frame origin.

layer
A layer can be

- A visible embedded image.
- The background that is displayed before the first image in the entire datastream is displayed. Its contents can be defined by the application or by the BACK chunk.

An embedded visible PNG or JNG datastream generates a single layer, even though it might be interlaced or progressive.

MNG-VLC
A very-low-complexity version of MNG in which only image 0 is permitted, loops can all be run once, JNG and Delta-PNG are not present, and certain MNG chunks are not present. Bit 0 of the MHDR simplicity profile must be 1, indicating that the profile is valid, and all other bits of the profile except possibly for bit 3 must be 0, indicating that “simple MNG features”, “complex MNG features”, JNG, and Delta-PNG are not present. Bit 3 can be 0 or 1, to indicate whether transparency is absent or present. MNG-VLC is a proper subset of MNG.

visible image
Actually drawn on a display. If an object is visible, a person looking at the display can see it.

3 MNG Chunks

This chapter describes chunks that can appear at the top level of a MNG datastream.

3.1 Critical MNG control chunks

This section describes critical MNG control chunks that MNG-VLC-compliant decoders must recognize and process (“processing” a chunk sometimes can consist of simply recognizing it and ignoring it; some chunks have been declared to be critical only to prevent them from being relocated by MNG editors).

3.1.1 MHDR MNG datastream header

The MHDR chunk is always first in all MNG datastreams except for those that consist of a single PNG or JNG datastream with a PNG or JNG signature.

The MHDR chunk contains exactly 28 bytes:

```
Frame width:  4 bytes (unsigned integer).
Frame height: 4 bytes (unsigned integer).
Ticks per
 second:      4 bytes (unsigned integer).
Nominal layer
 count:       4 bytes (unsigned integer).
Nominal frame
 count:       4 bytes (unsigned integer).
Nominal play
 time:        4 bytes (unsigned integer).
Simplicity
```

```
profile:     4 bytes:(unsigned integer).
               bit 0:
                      1: Presence or absence of certain
                         features is specified by the
                         remaining bits of the simplicity
                         profile.
                      (must be 1 in MNG-VLC datastreams)
               bit 1:
                      0: Simple MNG features are absent.
                      (must be 0 in MNG-VLC datastreams)
               bit 2:
                      0: Complex MNG features are absent.
                      (must be 0 in MNG-VLC datastreams)
               bit 3:
                      0: transparency is absent or can be
                         ignored.
                      1: transparency is present and is
                         essential (critical) for proper
                         display of the images.
               bit 4:
                      0: JNG is absent.
                      1: JNG is present.
                      (must be 0 in MNG-VLC datastreams)
               bit 5:
                      0: Delta-PNG is absent.
                      (must be 0 in MNG-VLC datastreams)
               bits 6-15:
                      Reserved for public expansion.  Must
                      be zero in this version.
               bits 16-30:
                      Available for private or experimental
                      expansion.  Undefined in this version
                      and can be ignored.
               bit 31:
                      Must be zero.
```

Decoders can ignore the “informative” frame-count, layer-count, play-time, and simplicity-profile fields.

The `frame_width` and `frame_height` fields give the intended display size (measured in pixels) and provide clipping boundaries.

The `ticks_per_second` field gives the framing rate. It must be nonzero if the datastream contains an animation. When the datastream contains exactly one frame, this field should be set to zero.

When `ticks_per_second` is nonzero, viewers should display animations at the rate of one frame per tick.

If the frame-count field contains a zero, the frame count is unspecified. If it is nonzero, it contains the number of frames that would be displayed, ignoring the TERM chunk. If the frame count is greater than $2^{31} - 1$, encoders should write $2^{31} - 1$, representing an infinite frame count. In MNG-VLC datastreams, the frame count is the same as the number of embedded images in the datastream.

If the layer-count field contains a zero, the layer count is unspecified. If it is nonzero, it contains the number of layers in the datastream, ignoring the TERM chunk. If the layer count is greater than $2^{31} - 1$, encoders should write $2^{31} - 1$, representing an infinite layer count. In MNG-VLC datastreams, the layer count is the frame count, plus one (for the background layer).

If the nominal-play-time field contains a zero, the nominal play time is unspecified. Otherwise, it gives the play time, in ticks, when the file is displayed ignoring the TERM chunk. Authors who write this field should choose a value of "ticks_per_second" that will allow the nominal play time to be expressed in a four-bit integer. If the nominal play time is greater than $2^{31} - 1$ ticks, encoders should write $2^{31} - 1$, representing an infinite nominal play time. In MNG-VLC datastreams, the nominal play time is the same as the frame count, except when the ticks_per_second is zero, in which case the nominal play time is also zero.

When bit 0 of the simplicity profile is zero, the simplicity (or complexity) of the MNG datastream is unspecified, and *all* bits of the simplicity profile must be zero. The simplicity profile must be nonzero in MNG-VLC datastreams.

If the simplicity profile is nonzero, it can be regarded as a 32-bit profile, with bit 0 (the least significant bit) being a "profile-validity" flag, bit 1 being a "simple MNG" flag, bit 2 being a "complex MNG" flag, bit 3 being a "transparency" flag, bit 4 being a "JNG" flag, and bit 5 being a "Delta-PNG" flag. The upper 15 bits (except for the most significant bit, which must be zero) are available for private test or experimental versions, and the remaining bits are reserved for future MNG versions, and must be zero in this version. If a bit is zero, the corresponding feature is guaranteed to be absent, and if a bit is one, the corresponding feature may be present in the MNG datastream.

"Transparency is absent or can be ignored" means that the PNG tRNS chunk is not present and no PNG or JNG image has an alpha channel, or that if they are present they are not essential (or critical) for displaying the images.

A MNG-VLC (i.e., a "very low-complexity MNG") datastream must have a simplicity profile with bit 0 equal to 1 and all other bits except possibly for bit 3 (transparency) equal to zero. If bit 4 (JNG) is 1, the datastream is a "MNG-VLC with JNG" datastream. MNG-VLC decoders are allowed to reject such datastreams unless they have been enhanced with JNG capability.

Encoders that write a nonzero simplicity profile should endeavor to be accurate, so that decoders that process it will not unnecessarily reject datastreams. For example, the simplicity profile 31 indicates that JNG, critical transparency, and at least one "complex" MNG feature are all present, but Delta-PNG is not. This example would not qualify as a MNG-VLC datastream because a "complex" MNG feature might be present. If the simplicity profile promises that certain features are absent, but they are actually present in the MNG datastream, the datastream is invalid.

3.1.2 MEND End of MNG datastream

The MEND chunk's data length is zero. It signifies the end of a MNG datastream.

3.1.3 LOOP, ENDL Define a loop

The LOOP chunk can be ignored by MNG-VLC decoders, along with the ENDL chunk.

3.2 Critical MNG image defining chunks

3.2.1 IHDR, PNG chunks, IEND

A PNG (Portable Network Graphics) datastream.

See the PNG specification [PNG] and the Extensions to the PNG Specification document [PNG-EXT] for the format of the PNG chunks.

Any chunks between IHDR and IEND are written and decoded according to the PNG specification.

The PNG oFFs and pHYs chunks and any future chunks that attempt to set the pixel dimensions or the drawing location must be ignored by MNG viewers and simply copied (according to the copying rules) by MNG editors.

The PNG gIFg, gIFt, and gIFx chunks must be ignored by viewers and must be copied according to the copying rules by MNG editors.

3.2.2 JHDR, JNG chunks, IEND

A JNG (JPEG Network Graphics) datastream.

See the JNG specification for the format of the JNG datastream.

Any chunks between JHDR and IEND are written and decoded according to the JNG specification.

The remaining discussion in the previous paragraph about PNG datastreams also applies to JNG datastreams.

MNG-VLC applications are not expected to process JNG datastreams unless they have been enhanced with JNG capability.

3.3 Critical MNG image displaying chunks

3.3.1 BACK Background

The BACK chunk suggests a background color against which transparent or less-than-full-frame images can be displayed.

```
Red_background:   2 bytes (unsigned integer).

Green_background: 2 bytes (unsigned integer).

Blue_background:  2 bytes (unsigned integer).

Mandatory
  background:      1 byte (unsigned integer).
                      0: Background color is advisory.
                         Applications can use it if they
                         choose to.
                      1: Background color is mandatory.
                         Applications must use it.
                      This byte can be omitted.  If so,
                      the background color is advisory.
```

Viewers are expected to composite the first layer in the MNG-VLC datastream against a background layer that fills the entire frame. The BACK chunk provides a background that the viewer can use for this purpose. If it is not “mandatory” the viewer can choose another background if it wishes. If the BACK chunk is not present, the viewer must provide its own background for the first subframe. Each layer after the first must be composited over the layers that precede it.

The three BACK components are always interpreted in the current color space as defined by any top-level gAMA, cHRM, iCCP, sRGB chunks that have appeared prior to the BACK chunk in the MNG datastream. If no such chunks appear, the color space is unknown.

Multiple instances of the BACK chunk are permitted in a MNG datastream.

The BACK chunk can be omitted. If a background is needed and the BACK chunk is omitted, then the viewer must supply its own background. For the purpose of counting layers, such a viewer-supplied background layer is counted the same as a background supplied by the BACK chunk.

In practice, most applications that use MNG as part of a larger composition should ignore the BACK data if `mandatory_background=0` and the application already has its own background definition. This will frequently be the case in World Wide Web pages, to achieve nonrectangular transparent animations displayed against the background of the page.

3.3.2 TERM Termination action

The TERM chunk suggests how the end of the MNG datastream should be handled, when a MEND chunk is found. It contains either a single byte or ten bytes:

```
Termination
  action:     1 byte (unsigned integer)
                0: Show the last frame indefinitely.
                1: Cease displaying anything.
                2: Show the first frame after the TERM
```

```
                   chunk.
                3: Repeat the animation starting
                   immediately after the TERM chunk.

Action after
  iterations: 1 byte
                0: Show the last frame indefinitely after
                   iteration_max iterations have been
                   done.
                1: Cease displaying anything.
                2: Show the first frame after the TERM
                   chunk.

               This and the remaining fields must be
               present if termination_action is 3, and
               must be omitted otherwise.

Delay:         4 bytes (unsigned integer).  Delay, in
               ticks, before repeating the animation.

Iteration max: 4 bytes (unsigned integer).  Maximum
               number of times to repeat the animation.
```

The TERM chunk, if present, must appear either immediately after the MHDR chunk or immediately prior to a SEEK chunk. Only one TERM chunk is permitted in a MNG datastream.

Simple viewers and single-frame viewers can ignore the TERM chunk. It has been made critical only so MNG editors will not inadvertently relocate it.

3.4 SAVE and SEEK chunks

Simple decoders that only read MNG datastreams sequentially can safely ignore the SAVE and SEEK chunks.

3.5 Ancillary MNG chunks

This section describes ancillary MNG chunks. MNG-compliant decoders are not required to recognize and process them.

3.5.1 eXPI Export image

The eXPI chunk takes a snapshot of an image, associates the name with that snapshot, and makes the name available to the "outside world" (like a scripting language).

The chunk contains an object identifier (snapshot id) and a name:

```
Snapshot id:   2 bytes (unsigned integer).
               (must be zero in MNG-VLC
               datastreams)
Snapshot name: 1-79 bytes (Latin-1 text).
```

When the snapshot_id is zero, the snapshot is the first instance of an embedded image following the eXPI chunk.

Note that the `snapshot_name` is associated with the snapshot, not with the `snapshot_id` nor its future contents; changing the image identified by `snapshot_id` will not affect the snapshot. The `snapshot_name` means nothing inside the scope of the MNG-VLC specification. If two eXPI chunks use the same name, it is the outside world's problem (and the outside world's prerogative to regard it as an error). It is recommended, however, that the `snapshot_name` not be the same as that appearing in any other eXPI chunk. A decoder that knows of no "outside world" can simply ignore the eXPI chunk. This chunk could be used in MNG datastreams that define libraries of related images, rather than animations, to allow applications to extract images by their `snapshot_id`.

Names beginning with the word "thumbnail" are reserved for snapshot images that are intended to make good icons for the MNG. Thumbnail images are regular PNG images, but they would normally have smaller dimensions and fewer colors than the MNG frames.

The `snapshot_name` string must follow the format of a tEXt keyword: It must consist only of printable Latin-1 characters and must not have leading or trailing blanks, but can have single embedded blanks. There must be at least one and no more than 79 characters in the keyword. Keywords are case-sensitive. There is no null byte terminator within the `snapshot_name` string, nor is there a separate null byte terminator. Snapshot names should not begin with the case-insensitive strings "CLOCK(", "FRAME(", or "FRAMES(" which are reserved for use in URI queries and fragments.

Multiple instances of the eXPI chunk are permitted in a MNG datastream, and they need not have different values of `snapshot_id`.

3.5.2 pHYs Physical pixel size

The MNG pHYs chunk is identical in syntax to the PNG pHYs chunk. It applies to complete MNG layers and not to the individual images within them.

The MNG top-level pHYs chunk can be nullified by a subsequent empty pHYs chunk appearing in the MNG top level.

3.5.3 Ancillary PNG chunks

The namespace for MNG chunk names is separate from that of PNG. Only those PNG chunks named in this paragraph are also defined at the MNG top level. They have exactly the same syntax and semantics as when they appear in a PNG datastream:

- iTXt, tEXt, zTXt
- tIME Same format as in PNG. Can appear at most once in the prologue segment (before the first SEEK chunk), and at most once per segment (between two consecutive SEEK chunks). In the prologue it indicates the last time any part of the MNG was modified. In a regular segment (between SEEK chunks), it indicates the last time that segment was modified.

A MNG editor that writes PNG datastreams should not include the top-level iTXt, tEXt, tIME, and zTXt chunks in the generated PNG datastreams.

The following PNG chunks are also defined at the MNG top level. They provide default values to be used in case they are not provided in subsequent PNG datastreams. Any of these chunks can be nullified by the appearance of a subsequent empty chunk with the same chunk name. Such empty chunks are not legal PNG or JNG chunks and must only appear in the MNG top level.

- cHRM, gAMA, iCCP, sRGB

A MNG editor that writes PNG or JNG datastreams is expected to include the top-level cHRM, gAMA, iCCP, and sRGB chunks in the generated PNG or JNG datastreams, if the embedded image does not contain its own chunks that define the color space. When it writes the sRGB chunk, it should write the gAMA chunk (and perhaps the cHRM chunk), in accordance with the PNG specification, even though no gAMA or cHRM chunk is present in the MNG datastream.

4 The JPEG Network Graphics (JNG) Format

JNG (JPEG Network Graphics) is the lossy sub-format for MNG objects. It is described in the full MNG specification and is also available as a separate extract from the full MNG specification. Both documents are available at the MNG home page,

`http://www.cdrom.com/pub/mng/`

MNG-VLC applications can choose to support JNG or not. Those that do not can check bit 4 (JNG is present/ absent) of the MHDR simplicity profile to decide whether they can process the datastream.

5 Chunk Copying Rules

The chunk copying rules for MNG are similar to those in PNG. Authors of MNG editing applications should consult the full MNG specification for details.

6 Minimum Requirements for MNG-VLC-Compliant Viewers

This section specifies the minimum level of support that is expected of MNG-VLC-compliant decoders, and provides recomendations for viewers that will support slightly more than the minimum requirements. All critical chunks must be recognized, but some of them can be ignored after they have been read and recognized. Ancillary chunks can be ignored, and do not even have to be recognized.

Applications that provide less than minimal MNG support should check the MHDR "simplicity profile" for the presence of features that they are unable to support. A specific subset, in which "complex MNG features", "simple MNG features" and JNG are absent, is called *"MNG-VLC"*. In MNG-VLC datastreams, bit 0 of the simplicity profile must be 1 and bits 1, 2, and 4 must be 0.

Subsets are useable when the set of MNG datastreams to be processed is known to be (or is very likely to be) limited to the feature set in MNG-VLC. Limiting the feature set in a widely-deployed WWW browser to anything less than MNG with 8-bit JNG support would be highly inappropriate.

Some subsets of MNG support are listed in the following table, more or less in increasing order of complexity.

Level of support	MHDR Profile bits								Profile
	31	30-6	5	4	3	2	1	0	value
MNG-VLC without transparency	0	0	0	0	0	0	0	1	1
MNG-VLC	0	0	0	0	1	0	0	1	9
MNG-VLC with JNG	0	0	0	1	1	0	0	1	25

One reasonable path for an application developer to follow might be to develop and test the application at each of the following levels of support in turn:

1. MNG-VLC,
2. MNG-LC (according to the MNG or MNG-LC specification),
3. MNG-LC with JNG,
4. MNG (according to the full MNG specification).

An equally reasonable development path might be

1. MNG-VLC with JNG,
2. MNG-LC with JNG (according to the MNG or MNG-LC specification),
3. MNG (according to the full MNG specification).

On the other hand, a developer working on an application for storing multi-page fax documents might have no need for more than "MNG-VLC without transparency".

6.1 Required MNG chunk support

MHDR

The ticks_per_second must be supported by animation viewers. The simplicity profile, frame count, layer count, and nominal play time can be ignored. Decoders that provide less than minimal support can use the simplicity profile to identify datastreams that they are incapable of processing.

MEND

The MEND chunk must be recognized but does not require any processing other than completing the last frame.

BACK

Must be fully supported.

DEFI, LOOP, ENDL, SAVE, SEEK, TERM

Must be recognized but can be ignored.

6.2 Required PNG chunk support

IHDR, PLTE, IDAT, IEND

All PNG critical chunks must be fully supported. All values of color_type, bit_depth, compression_method, filter_method and interlace_method must be supported (interlacing, as in PNG, need not necessarily be displayed on-the-fly; the image can be displayed after it is fully decoded). The alpha-channel must be supported, at least to the degree that fully opaque pixels are opaque and fully transparent ones are transparent. It is recommended that alpha be fully supported. Alpha can be ignored if bit 3 of the simplicity profile is 0.

tRNS

The PNG tRNS chunk, while it is an ancillary chunk, must be supported in MNG-compliant viewers, at least to the degree that fully opaque pixels are opaque and fully transparent ones are transparent. It is recommended that alpha data from the tRNS chunk be fully supported in the same manner as alpha data from an RGBA image or a JNG with an alpha channel contained in IDAT chunks. The tRNS chunk can be ignored if bit 3 of the simplicity profile is 0.

Other PNG ancillary chunks

Ancillary chunks other than PNG tRNS can be ignored, and do not even have to be recognized.

Color management

It is highly recommended that decoders support at least the gAMA chunk to allow platform-independent color rendering.

6.3 Optional JNG chunk support

Bit 4 of the simplicity profile can be used to promise that JNG chunks are not present. Viewers that choose not to support JNG can check this bit before deciding to proceed. MNG-VLC decoders are not required to support JNG.

JHDR, JDAT, IDAT, JSEP, IEND
: All JNG critical chunks must be fully supported. All values of `color_type`, `bit_depth`, `compression_method`, `filter_method` and `interlace_method` must be supported (interlacing, as in PNG, need not necessarily be displayed on-the-fly; the image can be displayed after it is fully decoded). The alpha-channel must be supported, at least to the degree that fully opaque pixels are opaque and fully transparent ones are transparent. It is recommended that alpha be fully supported.

JNG ancillary chunks
: All JNG ancillary chunks can be ignored, and do not even have to be recognized.

JNG image sample depth
: Only `image_sample_depth=8` must be supported. The JSEP chunk must be recognized and must be used by minimal decoders to select the eight-bit version of the image, when both eight-bit and twelve-bit versions are present, as indicated by `image_sample_depth=20` in the JHDR chunk. When `image_sample_depth=12`, minimal decoders are not obligated to display anything. Such decoders can choose to display nothing or an empty rectangle of the width and height specified in the JHDR chunk.

7 Recommendations for Encoders

The following recommendations do not form a part of the specification.

7.1 Use a common color space

It is a good idea to use a single color space for all of the layers in an animation, where speed and fluidity are more important than exact color rendition. This is best accomplished by defining a single color space at the top level of MNG, using and either an sRGB chunk or the gAMA and cHRM chunks and perhaps the iCCP chunk, and removing any color space chunks from the individual images after converting them to the common color space.

When the encoder converts all images to a single color space before putting them in the MNG datastream, this will allow decoders to improve the speed and consistency of the display.

For single-frame MNG datastreams, however, where decoding speed is less important and exact color rendition might be more important, it is best to leave the images in their original color space, as recommended in the PNG specification, to avoid any loss of data due to conversion, and to retain the individual color space chunks if the images have different color spaces.

8 Recommendations for Decoders

8.1 Decoder handling of fatal errors

When a fatal error is encountered, such as a bad CRC or an unknown critical MNG chunk, minimal viewers should simply abandon the MNG datastream.

8.2 Decoder handling of interlaced images

Decoders are required to be able to interpret datastreams that contain interlaced PNG images, but are only required to display the completed frames; they are not required to display the images as they evolve. Viewers that are decoding datastreams coming in over a slow communication link might want to do that, but MNG authors should not assume that the frames will be displayed in other than their final form.

8.3 Decoder handling of palettes

When a PLTE chunk is received, it only affects the display of the PNG datastream that includes it. Decoders must take care that it does not retroactively affect anything that has already been decoded.

If a frame contains two or more images, the PLTE chunk in one image does not affect the display of the other.

A composite frame consisting only of indexed-color images should not be assumed to contain 256 or fewer colors, since the individual palettes do not necessarily contain the same set of colors.

8.4 Behavior of single-frame viewers

Viewers that can only display a single frame must display the first frame that they encounter.

8.5 Clipping

MNG-VLC provides the following types of clipping, in addition to any clipping that might be required due to the physical limitations of the display device.

Frame width and frame height

The `frame_width` and `frame_height` are defined in the MHDR chunk and cannot be changed by any other MNG chunk.

This is the only type of clipping available in MNG-VLC datastreams.

Decoders can use these parameters to establish the size of a window in which to display the MNG frames. When the `frame_width` or `frame_height` exceeds the physical dimensions of the display hardware, the contents of the area outside those dimensions is undefined. If a viewer chooses, it can create “scroll bars” or the like, to enable persons to pan and scroll to the offscreen portion of the frame. If this is done, then the viewer is responsible for maintaining and updating the offscreen portion of the frame.

In the case of a MNG datastream that consists of a PNG or JNG datastream, with the PNG or JNG signature, the `frame_width` and `frame_height` are defined by the `width` and `height` fields of the IHDR (or JHDR) chunk.

9 Miscellaneous Topics

9.1 File name extension

On systems where file names customarily include an extension signifying file type, the extension `.mng` is recommended for MNG (including MNG-VLC) files. Lowercase `.mng` is preferred if file names are case-sensitive. The extension `.jng` is recommended for JNG files.

10 Revision History

10.1 Version 0.95a

Released 27 May 1999.

- Started a Revision History section.

10.2 Version 0.95

Initial public release, approved by the PNG Development Group on 11 May 1999.

11 Security Considerations

Security considerations are addressed in the PNG specification.

Some people may experience epileptic seizures when they are exposed to certain kinds of flashing lights or patterns that are common in everyday life. This can happen even if the person has never had any epileptic seizures. All graphics software and file formats that support animation and/or color cycling make it possible to encode effects that may induce an epileptic seizure in these individuals. It is the responsibility of authors and software publishers to issue appropriate warnings to the public in general and to animation creators in particular.

No known additional security concerns are raised by this format.

12 Appendix: Examples

We use the "#" character to denote commentary in these examples; such comments are not present in actual MNG datastreams.

12.1 Example 1: A single image

The simplest MNG datastream is a single-image PNG datastream. The simplest way to create a MNG from a PNG is:

```
copy file.png file.mng
```

The resulting MNG file looks like:

```
\211 P N G \r \n ^z \n  # PNG signature.
sRGB 2
IHDR 720 468 8 0 0 0 0  # Width and Height, etc.
IDAT ...
IEND
```

If `file.png` contains an sRGB chunk and also gAMA and cHRM chunks that are recommended in the PNG specification for "fallback" purposes, you can remove those gAMA and cHRM chunks from `file.mng` because any MNG viewer that processes the gAMA chunk is also required to recognize and process the sRGB chunk, so those chunks will always be ignored. Any MNG editor that converts the MNG file back to a PNG file is supposed to insert the recommended gAMA and cHRM chunks.

12.2 Example 2: A very simple movie

This example demonstrates a very simple movie, such as might result from directly converting an animated GIF that contains a simple series of full-frame images:

```
\212 M N G \r \n ^z \n  # MNG signature.
MHDR 256 300   # Width and height.
     1         # 1 tick per second.
     7 6 6     # Layers, frames, play time
     1         # MNG-VLC simplicity
TERM 3 0 120 10   # When done, repeat animation 10 times.

IHDR ...  IDAT ...  IEND # Six PNG datastreams
IHDR ...  IDAT ...  IEND # are read and displayed.
IHDR ...  IDAT ...  IEND
IHDR ...  IDAT ...  IEND
IHDR ...  IDAT ...  IEND
```

```
IHDR ...  IDAT ...  IEND
MEND
```

12.3 Examples 3-15: Omitted from MNG-VLC.

These examples in the full MNG specification use features that are not available in MNG-VLC.

12.4 Example 16: Storing an icon library

Here is an example of storing a library of icons in a MNG-VLC datastream. All of the icons use the same “sRGB” colorspace, so a global sRGB chunk appears at the beginning. The eXPI chunks allow applications to extract the images by name.

```
MHDR 96 96 1 6 5 5 9   # Profile 9 is MNG-VLC
sRGB 2                 # Global sRGB
eXPI 0 "thumbnail"
IHDR 32 32 ... PLTE ... tRNS 0 IDAT ... IEND
eXPI 0 "left arrow"
IHDR 96 96 ... PLTE ... tRNS 0 IDAT ... IEND
eXPI 0 "right arrow"
IHDR 96 96 ... PLTE ... tRNS 0 IDAT ... IEND
eXPI 0 "up arrow"
IHDR 96 96 ... PLTE ... tRNS 0 IDAT ... IEND
eXPI 0 "down arrow"
IHDR 96 96 ... PLTE ... tRNS 0 IDAT ... IEND
MEND
```

13 Credits

Editor

- Glenn Randers-Pehrson randeg @ alum.rpi.edu

Contributors

Contributors' names are presented in alphabetical order:

- Mark Adler, madler @ alumni.caltech.edu
- Thomas Boutell, boutell @ boutell.com
- John Bowler, jbowler @ acm.org
- Christian Brunschen, cb @ df.lth.se
- Glen Chapman, glenc @ clark.net
- Adam M. Costello, amc @ cs.berkeley.edu
- Lee Daniel Crocker, lee @ piclab.com
- Peter da Silva, peter @ starbase.neosoft.com
- Andreas Dilger, adilger @ enel.ucalgary.ca
- Oliver Fromme, oliver @ fromme.com
- Jean-loup Gailly, jloup @ gzip.org
- Chris Herborth, chrish @ qnx.com
- Alex Jakulin, Aleks.Jakulin @ snet.fri.uni-lj.si
- Gerard Juyn, gjuyn @ xs4all.nl
- Neal Kettler, neal @ westwood.com
- Tom Lane, tgl @ sss.pgh.pa.us
- Alexander Lehmann, alex @ hal.rhein-main.de
- Chris Lilley, chris @ w3.org
- Dave Martindale, davem @ cs.ubc.ca
- Carl Morris, msftrncs @ htcnet.com
- Owen Mortensen, ojm @ acm.org

- Josh M. Osborne, stripes @ va.pubnix.com
- Keith S. Pickens, ksp @ rice.edu
- Glenn Randers-Pehrson, randeg @ alum.rpi.edu
- Nancy M. Randers-Pehrson, randeg @ alum.rpi.edu
- Greg Roelofs, newt @ pobox.com
- Willem van Schaik, willem @ schaik.com
- Guy Schalnat, gschal @ infinet.com
- Paul Schmidt, pschmidt @ photodex.com
- Smarry Smarasderagd, smar @ reptiles.org
- Thomas R. Tanner, ttehtann @ argonet.co.uk
- Guido Vollbeding, guivol @ esc.de
- Tim Wegner, twegner @ phoenix.net
- Alaric B. Williams, png @ abwillms.demon.co.uk

Trademarks

- GIF is a service mark of CompuServe Incorporated.

Document source

This document was built from the file `mng-master-19990527` on 27 May 1999.

Copyright Notice

End of MNG-VLC Specification.