Overview

Animated Portable Network Graphic is a file format similar to an animated GIF. The Portable Network Graphic Animator will construct an APNG file from a collection of PNG image files and XML meta data. The XML entries will be provided by the user. The meta data will be combined with the PNG image data to construct an APNG file that can be viewed using the Mozilla Firefox web browser. In addition, the Portable Network Graphic Animator will be able to accept an existing APNG file and convert its frames into individual PNG image files, along with XML data with the information for each frame. Overall, this will allow creation of new APNG animations as well as modification of existing APNG animations.

PNG and APNG Specifications

PNG Specifications

Structure of a very simple PNG file

89 50 4E 47 0D 0A 1A 0A PNG signature	IHDR Image header	IDAT Image data	IEND Image end

Source: http://en.wikipedia.org/wiki/APNG

IHDR (source):

IHDR Chunk defined as follows:

Width	4 bytes
Height	4 bytes
Bit depth	1 byte
Colour type	1 byte
Compression method	1 byte
Filter method	1 byte
Interlace method	1 byte

-Bit Depth and Color Type

Table 11.1 — Allowed combinations of colour type and bit depth

PNG image type	Colour type	Allowed bit depths	Interpretation
Greyscale	0	1, 2, 4, 8, 16	Each pixel is a greyscale sample
Truecolour	2	8, 16	Each pixel is an R,G,B triple
Indexed-colour	3	1, 2, 4, 8	Each pixel is a palette index; a **PLTE** chunk shall appear.
Greyscale with alpha	4	8, 16	Each pixel is a greyscale sample followed by an alpha sample.
Truecolour with alpha	6	8, 16	Each pixel is an R,G,B triple followed by an alpha sample.

-Compression Method (zlib)

The only compression method currently used in PNG is 0, or zlib deflate/inflate compression with a sliding window. This compression uses a combination of LZ77 compression with Huffman Coding.

zlib data format is as follows:

zlib compression method/flags code	1 byte
Additional flags/check bits	1 byte
Compressed data blocks	n bytes
Check value	4 bytes

-Filter Method

x the byte being filtered;

a the byte corresponding to x in the pixel immediately before the pixel containing x (or the byte immediately before x, when the bit depth is less than 8);

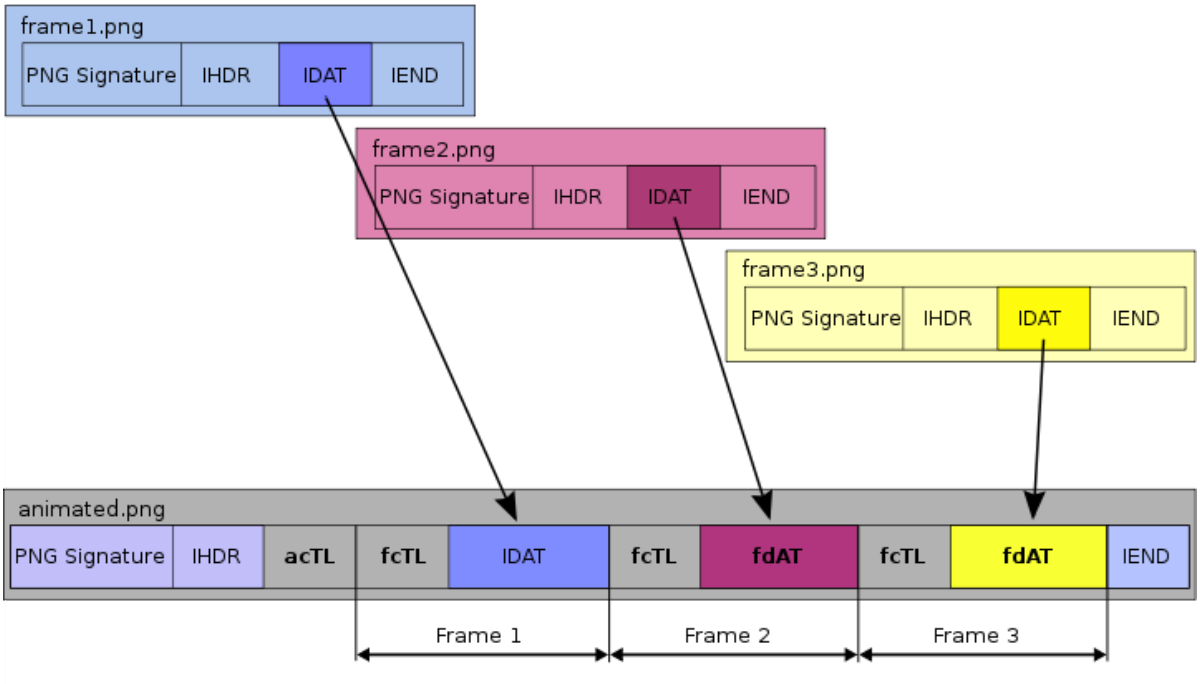
b the byte corresponding to x in the previous scanline;

c the byte corresponding to b in the pixel immediately before the pixel containing b (or the byte immediately before b, when the bit depth is less than 8).

Table 9.1 — Filter types

Type	Name	Filter Function	Reconstruction Function
0	None	Filt(x) = Orig(x)	Recon(x) = Filt(x)
1	Sub	Filt(x) = Orig(x) - Orig(a)	Recon(x) = Filt(x) + Recon(a)
2	Up	Filt(x) = Orig(x) - Orig(b)	Recon(x) = Filt(x) + Recon(b)
3	Average	Filt(x) = Orig(x) - floor((Orig(a) + Orig(b)) / 2)	Recon(x) = Filt(x) + floor((Recon(a) + Recon(b)) / 2)
4	Paeth	Filt(x) = Orig(x) - PaethPredictor(Orig(a), Orig(b), Orig(c))	Recon(x) = Filt(x) + PaethPredictor(Recon(a), Recon(b), Recon(c)

APNG Creation Diagram



Source: http://en.wikipedia.org/wiki/APNG#Technical_details

PNGA XML Schema

```
<pnga frames="5" plays="2">
        <image src="bob0.png" length="1/12"></image>
        <image src="bob1.png" length="1/14"></image>
        <image src="bob2.png" length="1/14"></image>
        <image src="bob3.png" length="1/14"></image>
        <image src="bob4.png" length="1/10"></image>
</pnga>
```

Modules

IO

main (apngxml, apngoutput, state) → Given an apngxml filename parses through the XML which adheres to the pnga XML schema to create an apng file which is saved to apngoutput. The animated image is composed of individual PNG frames referenced in the apngxml file along with their time offsets.

writeXML (frames plays imagesWithLength) → Given the appropriate information, writes an XML document for the newly created APNG.

writeAPNG (apngfiledata filename) → Given apng file data, writes the APNG file.

BasicLex – 154 Base LOC

(base from previous project)

split-at-delimiter (ds xs) → ds = delimiters to look for (list)

xs = object to search in (list)

(split-at-delimiter ds xs) = (before at+)

where before = longest prefix of xs with no values from ds (list)

at+ = rest of xs (list)

span (ps xs) → ps = list of signals to pass by (no constraints on signals)

xs = list of signals (no constraints on signals)

(span ps xs) = list of two elements

1 longest prefix of xs containing only signals from ps

2 rest of xs

splitoff-prefix (ps xs) → ps = prefix to look for (list)

xs = object to search in (list)

(splitoff-prefix ps xs) = (ps-matching ps-af-match xs-af-match)

where ps-matching = longest ps prefix matching xs prefix (list)

ps-af-match = rest of ps (list)

xs-af-match = non-matching suffix of xs (list)

Note: ps-af-match = nil means ps is a prefix of xs

splitoff-prefix-upr (ps xs) → ps = prefix to look for (list of standard, upper-case characters)

xs = object to search in (list)
(splitoff-prefix-upr ps xs) = (ps-matching ps-af-match xs-af-match)
where ps-matching = longest ps prefix matching xs prefix (list)
ps-af-match = rest of ps (list)
xs-af-match = non-matching suffix of xs (list)
Notes: 1. search is not sensitive to case of letters in xs
2. ps-af-match = nil means ps is a prefix of xs
Implementation issue: combining general and case-insensitive search
in one function simplifies maintenance, but complicates
formulation of software properties and their proofs

splitoff-prefix-chr (tok-str xs) → tok-str = characters to look for (string, standard characters)
chrs = object to search in

split-on-token-gen (tok xs) → tok = object to search for (list)
xs = object of search (list)
(split-on-token-gen tok xs) = (prefix match suffix)
where
prefix = elems of xs before 1st sublist matching tok (list)
= xs if no match
match = tok if match (list)
= nil if no match
suffix = elems of xs after 1st sublist matching tok (list)
= nil if no match

split-on-token-chr (tok xs) → tok = object to search for (list of upper-case standard characters)
xs = object to search in (list containing no non-standard chars)
Note: matching is not case-sensitive
(split-on-token-chr tok xs) = (prefix match suffix)
where
prefix = elems of xs before 1st sublist matching tok (list)
= xs if no match
match = tok if match (list)

= nil if no match

suffix = elems of xs after 1st sublist matching tok (list)

= nil if no match

split-on-token (tok xs) → tok = object to search for (string or list)

xs = object to search in (list, no non-standard chars if tok is string)

Note: search is not case-sensitive if tok is a string

Warning! Neither tok nor xs may contain non-standard characters

if tok is a string

Implementation issue: combining general and case-insensitive search

in one function simplifies maintenance, but complicates

formulation of software properties and their proofs

MinidomParser – 317 Base LOC

(base from previous project)

xml-getnodes (node nodename) → returns children of node with type nodename

xml-getdeepnodes (node nodename) → returns children of node with type nodename searching recursively using DFS with node as root.

xml-getnode (node nodename) → returns first child node with type nodename

xml-getdeepnode (node nodename) → returns first child node with type nodename searching recursively using DFS with node as root.

xml-getattribute (node attributename) → returns the value of node's attribute with name attributename

xml-gettext (node) → returns the composite of all text inside of a node

xml-isattribute (attribute) → returns true iff attribute is an mv of length 2 with both elements of the mv being strings

xml-isattributelist (attributes) → returns true iff attributes is nil or a list of attributes

xml-isnode (node) → returns true iff node is actually a node

xml-isnodelist (nodes) → returns true iff nodes is a list of nodes

MinidomSerializer – 125 Base LOC

(base from previous project)

xml-readnode (xmlchars) → returns the root node from xmlstring

xml-readnodes (xmlchars) → returns (mv nodes remainingxmlstring)

xml-unescape (escapedchars) → string with entities replaced

xml-readnodeproperties (xmlchars) → returns (mv attributes remainingxmlstring)

xml-skipdontcares (xmlchars) → returns next xmlchars sans don't cares

xml-escape (unescapedchars) → returns string with bad chars replaced with entities

xml-serizlize-dom (xmlnode) → Returns a string containing an xml document that represents the dom passed in through xmlnode.

xml-serialize-attributes (attributes) → Returns a string that is xml that represents the passed in attribute list.

xml-serizlize-nodes (xmlnodes) → Returns a string containing xml nodes that represents the node list, xmlnodes.

PNGFileParser

readChunk (pngfiledata) → Given pngfiledata which is a sequence of PNG data chunks, this function returns (mv chunktype chunkdata remainingdata) where chunktype is the type of chunk, chunk data is all the data contained in this chunk and remainingdata is the rest of the unparsed PNG chunks.

getWidth (chunkdata) → Given the chunk data of a IHDR chunk type returns the width.

getHeight (chunkdata) → Given the chunk data of a IHDR chunk type, returns the height.

getBitDepth (chunkdata) → Given the chunk data of a IHDR chunk type, returns the bit depth.

getColorDepth (chunkdata) → Given the chunk data of a IHDR chunk type, returns the color depth.

getCompMethod (chunkdata) → Given the chunk data of a IHDR chunk type, returns the compression method.

getFilterMethod (chunkdata) → Given the chunk data of a IHDR chunk type, returns the filter method.

getILMethod (chunkdata) → Given the chunk data of a IHDR chunk type, returns the interlace method.

readIHDR (chunkdata) → Given the chunkdata of a IHDR chunktype chunk returns the corresponding (mv width height bitdepth colordepth compressionmethod filtermethod interlacemethod) where the different values are those specified in the chunkdata.

PNGDatParser (Optional)

APNGBuilder

doChunk (chunkname, chunkdata) → Generic function to write any type of chunk to the given file.

doIHDR (width, height, bitdepth, colordepth, compressionmethod, filtermethod, interlacemethod) → Specialized function to write an IHDR chunk.

doACTL (numframes, numplays) → Specialized function to write an aCTL chunk.

doFCTL (sequencenum, width, height, xoffset, yoffset, delaynum, delayden, disposeop, blendop) → Specialized function to write an fCTL chunk.

writeFdAT (sequencenum, framenum) → Specialized function to write an fdAT chunk.

APNGExploder

explodeAPNG (apnginput, apngxml, pngnamebase, state) → Splits up the given APNG into its constituent frames, outputting them to files beginning with pngnamebase prepended to the frame sequence number. Additionally the XML configuration file is output to a file with the name given by apngxml.

chopFrame (apngfiledata) → Given apngfiledata, returns the data for the first frame found.

makePNG (framedata) → Given APNG frame data, returns PNG file data for an individual frame.

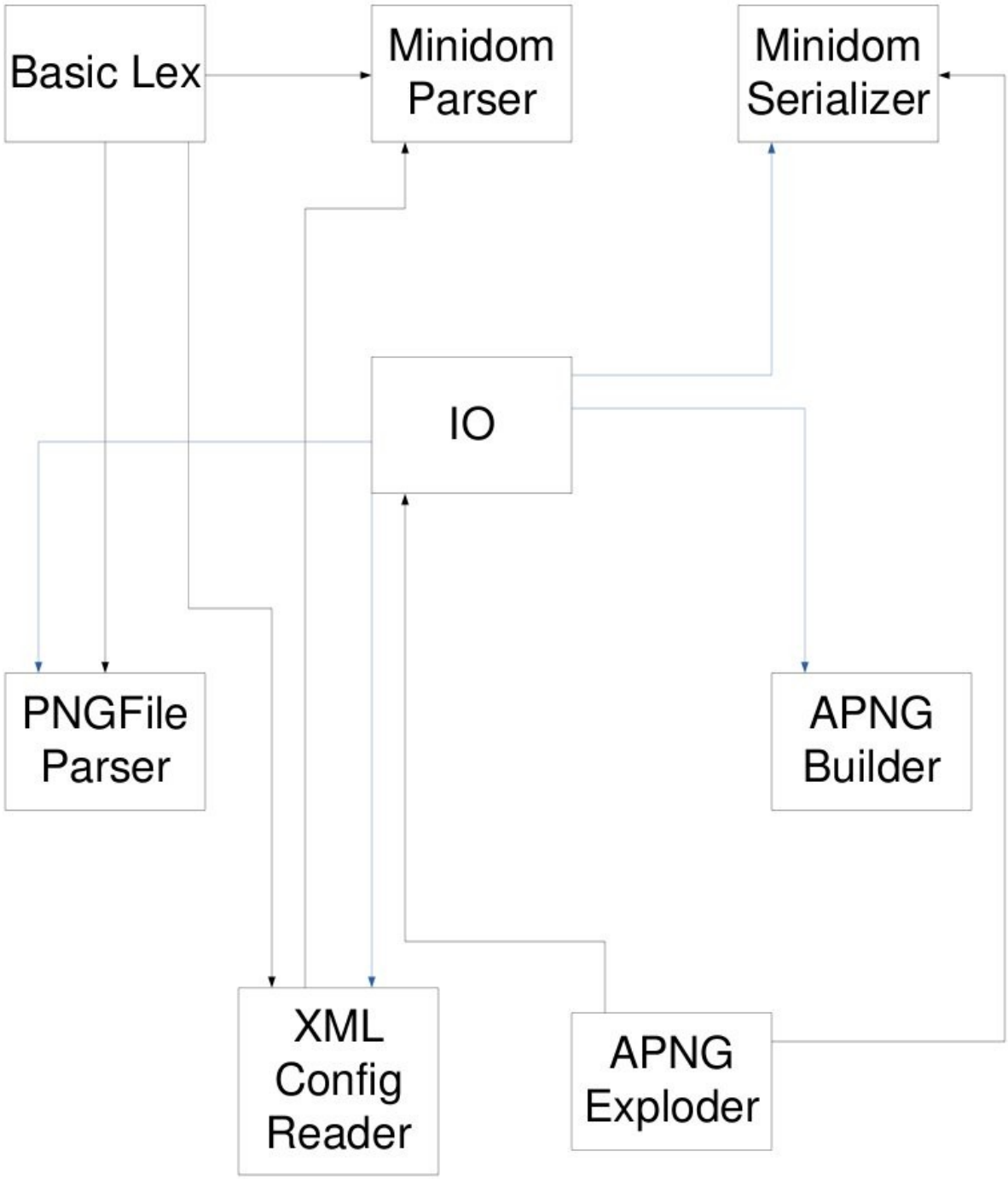
XMLConfigReader

numPlays (xml) → Given the xml data, returns the number of times the animation plays.

numFrames (xml) → Given the xml data, returns the number of frames in the animation.

getImg (xml) → Given the xml data, returns the first frame for the animation.

getLength (xml) → Given the xml data, returns the length for the first frame in the animation.


Basic Lex
Minidom Parser
Minidom Serializer
IO
PNGFile Parser
APNG Builder
XML Config Reader
APNG Exploder

PROBE Estimate

	Tiny (7%)	*Small(24%)*	*Medium(38%)*	*Large(24%)*	*Huge(7%)*
Avg. LOC	2	4	7	18	43

Base LOC:

MindomParser	317
MinidomSerializer	125
BasicLex	154

New LOC:

main(Huge)	43	doChunk(Huge)	43
readChunk(Large)	18	doIHDR(Large)	18
getWidth(Huge)	43	doACTL(Large)	18
getHeight(Huge)	43	doFCTL(Large)	18
getBitDepth(Huge)	43	writeFDAT(Huge)	43
getColorDepth(Huge)	43	explodeAPNG(Huge)	43
getCompMethod(Huge)	43	writeXML(Huge)	43
getFilterMethod(Huge)	43	writeAPNG(Large)	18
getILMethod(Huge)	43	chopFrame(Huge)	43
readIHDR(Large)	18	makePNG(Huge)	43
numPlays(Medium)	7	numFrames(Medium)	7
getImg(Medium)	7	getLength(Medium)	7

Contracts

IO – 2 Large cons – 36

PNGFIleParser – 9 Huge cons – 387

APNGBuilder – 5 Huge cons – 215

APNGExploder – 3 Huge cons – 129

XMLConfigReader – 4 Large cons - 72

Total Estimated LOC: 2137