Evan Hatton

Suzzane Rivoire

CS 496

February 12th, 2022

**Introduction**

The main driving inspiration for this project was my exposure to textual ASCII and ANSI art, especially the adaptation of Star Wars into a terminal animation viewable over telnet at towel.blinkenlights.nl, along with having a typewriter. I noticed there wasn’t really anybody treating the text as an image that could be freed from its normal default display of horizontal, left to right. I wanted to more or less elevate a file format that could capture the art I produced with a typewriter by making the paper feed in an angle or double striking characters in unaligned portions by manually pushing the carriage by a fraction of the regular striking distance.

My idea from the start was to elevate the importance of text into image level, with some sort of markup to style it over a rectangular frame. Then I realized it would be more interesting if the text could be placed over some sort of image by default. In order to get a solid grounding on the best ways to do this I did a tour of all the modern popularly used image and document formats.

In short my tour reveals that there is a lack in this space for current file types – the general formats that can accomplish this are insecure and lake widespread support for being treated as images – document style formats all assume these will be sent to a printer or similar 8.5”x11” profile. Power Point files are the most similar in idea to the specification I create. In the end Power Point are expected to be presented on a large screen in addition to the security vulnerabilities that along with a lack of viewing only support makes them not a direct replacement.

The file type I end up creating is called Extended Text. It steals the best elements from the files I toured and adds its own twists to try and avoid security vulnerabilities while still accomplishing the text layering.

# Current File Type Analysis

Going over what classifications of image formats exist currently, there are raster/bitmap formats, compressed bitmap, vector image and multilayered.

Uncompressed Bitmap

TIFF

This format is one of the few formats still in wide circulation that has the explicit option for uncompressed data built into its specification. The sixth revision of the specification was released as an open standard by Adobe 35 years ago[[1]](#footnote-0). It offers a baseline that all readers and editors must support to meet the specifications and then some optional extensions. Interestingly, built into the specification is a size cap at being a maximum of 232 which begs the question of whether current readers follow this particular portion, due to size limitations not being as much of a concern as they were in 2002. The main utility of the TIFF format is to be able to edit images and save them without having any loss of data, making them an ideal archival format that does not require specialized or paid software that the numerous types of RAW formats might need to edit and view. The readers necessarily must be prepared to deal with multiple sub images defined within the TIFF file but do not necessarily have to display the subfiles beyond the first. Additionally the images themselves are composed of one or more “strips” which are rectangular swaths of pixels. The image being split into strips allows for buffered viewing and, more importantly, each strip to use different compression schemes. The tag fields are also optional and readers must be prepared to handle them. These data tags were extended to the EXIF tagging we are familiar with today.

Compressed Formats

JPEG

First published in 1992, this lossy compressed 1image format in its current incarnation is widely used for its ability to compress images of real life areas to tiny data spaces while having almost zero to very tiny quality loss to the human eye. The original basis for this technique is based on a proposal in 1972 by Nashir Ahmed to encode images using a discrete cosine transform along with a few patents by IBM, Mitsubishi, AT&T and Canon. JPEG is actually an acronym that stands for Joint Photographic Experts Group which maintains the specification and codec for JPEG, JFIF (the actual file container format) and other offshoot specifications. There has been some legal controversy in the past fighting over the patent for JPEG but they have since expired and baseline JPEG is truly a free format now.

In my personal experience I have compressed images I wanted to keep that had been sent to me in PNG but were of nature or similar and around 6MB per image. Doing a quick switch to the JPEG format at maximum quality resulted in an image that had no noticeable difference and was about 200 KB – the PNG form is nearly 30 times the size. While JPEG has huge compression savings, the cost is that rapidly changing data is not preserved well. This flaw results in artifacts – squares of averaged color data – that lose a decent amount of information. In photography like images this is almost never a problem, even images that include writing (on say a sign) often have many pixels dedicated to each character. The compression of the high frequency data results in images where the text of say a screenshot of a computer screen or an image taken with a camera of a book is unreadable or very fuzzy on the edges, and if the quality is very low it might even result in the text disappearing.

The discrete cosine transform (DCT) is the main driver behind how the JPEG format works; however first the compression comes in from how the colors are represented before even being put through the DCT. Colors on a monitor are usually represented by the RGB color space, yet scientists have teased out that human eyes are much more sensitive to differences in brightness than differences in color. By using a different color encoding that elevates the importance of brightness the encoding can focus less on color information. The YCrCb color space is the one which was chosen. The encoding process can discard about half of the color information, called Chroma Subsampling, just with this technique with minimal effect to the final image with respect to human vision. The DCT works by getting sample points off of cosine waves that represent pixel patterns. Each JPEG image is split into a grid of 8x8 pixel squares, each block has its values between the values of -128 and 127. Then it takes each row through the DCT into a matrix and then each column into a different matrix, defining a 2D version of the DCT. Rebuilding the image with every single coefficient from these matrices results in an image nearly lossless from the original. Most of the information for an image is compacted into the first nine or so coefficients, which results in blocks that quickly reach high quality to the human eye with only a fraction of the 64 coefficients collected. The process of dropping coefficients is called Quantization. In implementation an 8 by 8 matrix for the whole image is generated based upon a user provided quality level. The Quantization matrix is then multiplied against every block and produces the final encoded coefficients. These coefficients are finally losslessly compressed using a zigzag run length encoding that compresses repeating consecutive digits to only two digits and then a Huffman encoding on those.

In relation to my overarching project, JPEG seems like a very difficult thing to implement correctly. With more time I could potentially adapt a freely licensed implementation. In the meantime, I believe that the features to tentatively earmark is the color space reduction into YCrCb, which can alone reduce space when compared to a PNG up to half.[[2]](#footnote-1), [[3]](#footnote-2)

PNG

Unlike the document formats or JPEG, PNG was introduced as a format specifically for sharing images over a network and PNG was intended to replace the GIF standard; the company which owned the rights to GIF wanted to start charging licensing fees circa 1994. In a coordinated effort, a group in 1995 started working to develop the eventual PNG specification and by 1997 PNG was in the track to be recognized as a technology standard in 2004[[4]](#footnote-3). The non-support of animations was because at the time the PNG standard was drafted, animated GIFs were very rare[[5]](#footnote-4). As another insight to how PNG is layed out, BMP has each point store the RGB values while PNG stores and compresses them based upon previous values. PNG has many features that put it as superior for applications in image editing or view on the web: first and foremost the lossless data compression makes it both smaller than if encoded in GIF while still not losing any important high frequency information that might be lost if it were encoded in a JPEG. The options for the color quality reaching up to 64 bits per pixel and a transparency channel makes PNG the default image format for most image editing applications[[6]](#footnote-5).

The file format of a PNG is split into chunks, some of which are optional. The non optional chunks are:

IHDR which defines the baseline characteristics of the image

IDAT the image data, which be in multiple IDAT chunks

PLTE which defines the colors used in the image

IEND marks end of image

The optional chunks can store information such as a default background color, exif data, annotation/caption text, the date that the image was last edited and other useful information that isn’t necessarily image data.

The actual compression part first does a filter pass to make the values better for the DEFLATE compressor, which is also the same compression algorithm used for ZIP files. The scheme used for the filter changes the value of the pixel based on previously seen values. In the PNG’s specification this can only be done on a line by line basis.

SVG

Stands for Scalable Vector Graphics and is an XML Markup style file format. Developed by the World Wide Web consortium, this is an open standard since 1999. The format supports interaction from the viewer and animation via JavaScript. Alternatively the SVG could also be animated through a timeline from the XML based SMIL that defines multimedia presentations or using CSS. Within the file there can be three elements, vector graphic objects, bitmap images and text.

There are a total of fourteen feature sets present in the SVG 2011 specification. Some of them are –

Paths: drawing lines including Beizer curves and other freeform lines.

Shapes: including arbitrary polygons, a straight line, straight line paths, stars, ellipse and circle.

Text: Unicode, some portions are handled via the XML specification. This element can be drawn along a curved path. Handles other typesetting attributes such as kerning, rotation etc.

Painting: Allows any enclosed area of an SVG to be painted in or outlined.

Color: sets the color of the inside or outline of any SVG element.

Gradients: can be a separate element or used to do a gradient fill or outline to any other element.

Clipping/Masking: allows use of SVG elements as filters for other elements, i.e. using a rectangle over a gradient and only showing inside that rectangle.

Filter Effect: Some kind of effect applied to an SVG element, such as emboss, metallic or so on.

Interactivity: Defines what user actions can be detected by the DOM.

SVG renderers are built into all of the major web browsers and many image editing applications. However, while SVG can be thought of as an image format, it really is XML and has security vulnerabilities, such as HTML injection (when viewed in a web browser) or cross-site scripting by embedding malicious links in the XML.

## Document Formats (Multilayered)

Starting with looking at plaintext, ASCII and especially Unicode came from a long line of evolution. ASCII was an encoding agreed upon that was originally a 7-bit coding of the Latin alphabet plus a few other important characters[[7]](#footnote-6). ASCII replaced the widespread use of the Baudot codes. Those codes were intended for usage of electronic typewriters in the telegraphic communication networks run by Western Union and the Teletypewriter Exchange Service (an AT&T subsidiary)[[8]](#footnote-7). Unicode used the 8th bit to extend to a large number of extra and optional character sets. This evolution is important because the document formatting programs stored their files in whatever operating system’s plaintext code was used at the time.

Following the earliest thread, shortly after ASCII was introduced, TJ-2 was released in 1963 as the first document typesetting program for printing[[9]](#footnote-8). Before this period the equivalent to TXT files had to be hand typeset for each specific printer. This program inspired improvements for MIT’s time sharing system (CSTT) which was released only about a year after in the form of TYPST and RUNOFF[[10]](#footnote-9). In Bell Labs the researchers were working on creating Unix and made a version of RUNOFF that ran on the system called roff[[11]](#footnote-10), which was later extending into troff and nroff[[12]](#footnote-11) around the year 1973. In 1977 Donald Knuth got inspiration to write a typesetting program to replicate the look of older printing press machines and released the first version of TeX in 1978[[13]](#footnote-12) which largely replaced troff, though nroff is still used for displaying man and info pages in Unix based systems. The most popular version of the TeX program is LaTeX, which was released much later in 1984[[14]](#footnote-13) by Leslie Lamport. The document specification system can save the source code in a .tex file and that specific code is compiled into a DVI file that can be directly printed or converted to a different document format, usually PDF.

In 1984 Adobe Systems released the first version of the PostScript[[15]](#footnote-14), a document description programming language. The motivation behind PostScript came for an improvement on the Xerox Interpress. Interpess was a page description format first released in 1978 written in the Forth programming language[[16]](#footnote-15). Two of the people who wrote Interpress left Xerox to found Adobe Systems and later wrote PostScript which is modeled after Interpress. PostScript represents everything besides rasterized images in a vectorized format that is independent of the printer. Often earlier computers that used PS as a GUI system. The GUIs of older applications often didn’t have the ability to accurately display a preview of PS documents – they weren’t as full features. Using PS to display a PostScript document makes the preview 100% accurate with the print version. Additionally within the Adobe ecosystem, PostScript is the backbone of a file type named encapsulated PostScript (eps), essentially an eps file defines a single rectangular region similarly to image files or an .svg instead of a printable document. The eps files are still in use within Adobe’s suite of software and have been excluded from being included in Microsoft’s Office Suite of programs because of security concerns. PS is a programming language whose programs render printable documents – they have been used to run malicious scripts[[17]](#footnote-16) to unwitting users. In 1992, Adobe released the PDF, based upon a former Camelot project; its goal was to allow for documents to be digitized and viewable digitally, rather than as descriptions for a printer to render a physical document[[18]](#footnote-17).

Meanwhile at IBM in 1970, a group of researchers at IBM defined a language for defining areas of text into categories called Generalized Markup Language[[19]](#footnote-18). This was a step up from plaintext because portions of the text separated logically into paragraphs, lists or titles, which let supporting printing programs format the document for the specific dot matrix printer without ever touching the document by hand. GML was refined and extended to Standardized GML (SGML) and became an international standard technology in 1986[[20]](#footnote-19) for documents and was widely used in government digitization of documents which needed to remain readable decades later. About ten years later SGML was extended to XML as its utility for representing data useful for web documents was recognized in 1996[[21]](#footnote-20). The first version of HTML is also heavily based upon the SGML variant that was used in CERN called CERNDOC[[22]](#footnote-21). XML is the base standard of the Open Document and Microsoft’s Office Open XML formats for the Office Suite. Similarly to the PostScript security concerns, these documents also have vulnerabilities in the macros and as of February of 2022 Microsoft has disabled running them by default.[[23]](#footnote-22), [[24]](#footnote-23)

In 1985 Electronic Arts and Commodore cooperated to create a file format to facilitate data transfer between software[[25]](#footnote-24). The goal was to make the machines more free and user conscious by creating the Interchange File Format that stores data and has essentially types stating what the data is. In 1998 a group of computer vision pioneers at AT&T Bell Labs released a paper describing their development of the DjVu technology[[26]](#footnote-25) which is notable in that it has a higher compression ratio for scanned documents than even JPEG images. DjVu’s point is separating out the data of scanned documents to achieve better compression for web viewing at still high quality. This format beats PDF files sizes at many scanned documents, by having separate OCR text layers and background image layers. Due to being a competitor for the PDF file format, DjVu eventually lost due to not enough accessibility tools existing initially and has slowly fallen out of usage. This format is the least common of the ones mentioned so far and out of the document formats presented is the last widely notable one before obscure very niche files pop up.

While the average office worker might encounter numerous types of non-obscure document format file extensions, all of them can be put into at most four different related families. The DjVu format gets its own category due to the origin and is probably more related to book archiving formats even if the file structure is different, such as EPUB or CBR, that are out of the scope of this analysis. PDF documents are all technically self contained PostScript programs that display a document when run with an interpreting viewer. The office suite of files, xlsx, ppt, docx etc, are all based upon XML, which is a language that allows defining a markup language and then using those definitions in the same document. These files use XML to state what part of the document is what and then the viewer displays it according to the rules for say a “Header” or a “Presentation Slide Bullet Point”. The LaTeX or TeX files, which an office worker would probably never see, are a type of markup language specifically designed to resemble “hot metal” typesetting along with extensions for complex mathematical formulas, diagrams or tables in a more accurate way than the XML documents might render them.

## PDF

(Portable Document Format) Inside the PDF, everything besides specific sections are represented as 7 bit ASCII characters. Some of the specific sections include image stream data. The internal structure revolves around a “Carousel Object Structure” that ostensibly PDF only makes use of a subset and COS is similar to both XML and JSON. COS has the following 9 objects:

Bools, Integers, Floats, Strings (these can have 8 bit characters), Names, Arrays (for ordering objects in a collection), Dictionaries, Streams and null.

These objects can be embedded within another object or “indirectly referenced” in a lookup table of already defined objects. The keyword xref, which is short for “cross reference” table. This table is always in the footer of the PDF file. This reusage of objects can significantly reduce file sizes. Due to that there is an extra CPU cost in looking up and building objects in the document, so there are a few optimizations to make loading times faster that consist of making objects on pages load sequentially starting from the first page. For example, this optimization makes it so that users don’t have to wait until the entire document loads in order to see an object appear on the first page that was defined on the last page.

PDF graphics are essentially designed the same as PostScript. There are 25 parameters that can be changed to alter the graphical elements. A subset of them are applied to a matrix to scale, rotate, and other transformations, to the elements independent of the machine. Images are put as Dictionary objects with their binary stream data in the dictionary in a form resembling the image format. The update to PDF 1.4 in 2001[[27]](#footnote-26) introduced transparency, which meant that elements could finally be drawn on top of each other, see various watermarks (ex: unofficial transcript).

Version 1.5, in 2003[[28]](#footnote-27), introduced the concept of Layers, which are COS dictionaries in the document that can be hidden or collapsed by authors and viewers. Version 1.2 from 1996[[29]](#footnote-28) is the version where form data was introduced. Additionally fonts can either state a typeface and hope that the user has it installed or embed the font into one of the COS objects and load it from the PostScript interpreter.

# ETXT (Extended Text)

This is my specification for a layered image format. After touring the current file types, nothing covers the use case that I am exploring. Some of the files currently can put text over the top of an image: one can currently use HTML, SVG, any of the office documents or PDF files to currently do that; all of the files which can layer text and images do not act like images apart from SVG. The work to maintain a PDF, SVG or document format viewer is significantly more work than a PNG or JPEG viewer. Additionally all of the layered formats have a history of security vulnerabilities when run natively. Having a separate file that can discard unnecessary features and be treated as an image rather than a document is better.

ETXT, like GIF, will have a window that may or may not be independent from the size of images inside of it. The display window is made up of one or more text fragments and zero or more images arranged in a pattern. The window also has multiple frames which define different arrangements for each image or text fragment. When fully implemented a user should be able to click on an image and that would change which frame is displayed by the renderer.

Being able to separate out the textual information like this brings with it many benefits over current image formats. One main selling point is that it can be used to deduplicate images that are heavily reused and the only difference between images is the text. This is most commonly seen within meme images. The deduplication saves bandwidth for both end users on limited data plans and saves costs for data center operators. Already, user’s web browsers are set up to cache files and serve them instead of redownloading images – an ETXT that takes the place of 1000 images can quickly add up across millions of requests. Another data saving usage is that the text can greatly increase the file size of a PNG file when placed over solid colors. Other benefits include serving the text to programs that aid the visually impared and losslessly preserving the images.

The following features for future versions are:

* Randomized frame progression
* Conversion to non-etxt image formats
* Text reader support

# Spec

The internals are based a little bit off of PDF – one line called the layout has the information for style and for what goes where in what style for each frame. Similar to the object table in a PDF.

An ETXT file must a minimum have these lines:

* layout – defines each frame and information for objects
* frameCode – this is a line for frameKey that transforms the numbers into the correct characters.
* frameKey – a line that states what the data in each frame of the layout object corresponds to. Frames are separated by a dash character.
* xCord – a list of x coordinates that are used in this file.
* yCord – a list of y coordinates used in this file.
* text – an underscore delimited text string. The unscores can be escaped by prepending an invisible bell characters (ascii character number 7)
* frameLookup – the only allowed dictionary. Space delimited text, where the first character is the key and the next contiguous text is the name of the data before the first space.
* defaults – currently stores the width and height of the window

Optional features are:

* Path – currently just straight lines. Each one is defined by a group of X1 Y1 X2 Y2 coordinates on the line.
* Color – html colors for text
* Size – font size
* Width – size of text outline
* b – color of text outline
* Font

Each frame in the frameKey must dedicate the first four entries in the specific order of “i x y t x y”, which means each frame has the coordinates of the images and text defined in the first six spots of the layout data for that frame. Optional styling for the text may be present afterwards.

The layout uses positional alignment to state what styling information is associated with which text fragment or image. So in frame one, image 0 corresponds to xCord 0 and yCord 0. The images are numbered from 0 in the order they are defined within the image. Then each data after that starts with the number of data points immediately after it.

Images are stored in binary, surrounded by triple layered opening curly braces, like so:

{{{

Image binary blob data

}}}

Embedded fonts are also stored in binary, surrounded by triple layered opened and closing parenthesis like so:

(((

Font binary blob data

)))

An example of a proposed ETXT file:

ETXT0.1

layout 2 6 0 1 2 2 2 3 6 0 1 10 14 10 15 6 0 3 5 5 16 15 16 0 1 2 3 2 4 5 4 6 7 8 9 10 11 12 13 16 12 11 11 19 18 -1 -1 -1 -1 7 8 9 6 16 17 13 16 1 2 4 2 4 -1 -1 -1 -1 9 13 19 23 10 14 21 16 -1 -1 -1 -1 -1 0 1 2 3 -1 -1 -1 -1 -1 -1 -1 16 3 0 0 0 0 0 0 0 0 1 1 1 1 1 1 2 16 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 2 16 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 0 0 0 0 0 3 0 1 2 3 0 1 10 3 0 3 5 18 0 21 22 21 23 4 5 4 6 11 12 14 15 18 19 20 16 17 18 12 8 8 14 14 -1 -1 -1 -1 7 8 20 19 11 9 7 20 13 18 1 3 4 3 4 -1 -1 -1 -1 9 13 8 13 18 19 23 18 22 18 -1 -1 -1 -1 -1 0 1 2 3 -1 -1 -1 -1 -1 -1 -1 -1 -1 18 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 18 1 0 0 0 0 0 0 0 0 0 0 2 2 2 0 0 2 2

frameCode i t x y p c b s w

frameKey 0 2 3 1 2 3 4 5 6 7 8 - 0 2 3 1 2 3 4 5 7

frameLookup i images t text p path x xCord y yCord c color b color s size w width

defaults 888 575

width 2

size 30 37 60

xCord 0 29 54 59 91 94 192 208 221 222 226 239 387 586 596 608 625 627 630 661 530

yCord 0 33 78 80 115 139 176 191 227 268 277 280 294 313 322 351 363 395 418 491 492 506 513 536 92 path 2 12 2 6 4 11 4 7 3 22 3 17 5 20 5 18

text\_The Gamer Matrix\_DON'T\_GET\_EXCITED\_GET\_GAME IS\_GOOD\_ARSE\_PLEASANTLY\_SURPRISED\_SMUG SELF\_SATISFACTION\_ALL IS\_WELL\_SUICIDE\_HOSTAGE\_IP\_POINT AND\_LAUGH\_RARE\_PITY WHEN\_THIS HAPPENS\_GAME DEVELOPER /\_PUBLISHER IS GOOD\_PUBLISHER IS ARSE

color #000000 #ffffff #fe0000 #fed30a

{{

iVBORw0KGgoAAAANSUhEUgAAA3gAAAI/AQMAAAAfiultAAAABlBMVEUGCgX+0wr7ypNZAAAA/klEQVR42u3dsQkAIAwAwYCFazu4GDt7QRTkvg7cAiGJkCRJkiRJkqRrlVy13KvXrWkej8fj8Xg8Ho/H4/F4PB6Px+PxeDwej8fj8XinPEkP9l8Gj8fj8Xg8Ho/H4/F4PB6Px+PxeDwej8fj8Xg8Ho/H4/F4PB6Px+PxeDwej8fj8Xg8Ho/H4/F4PB6Px+PxeDwej8fj8Xg8Ho/H4/F4PB6Px+PxfvIk+f/A4/F4PB6Px+PxeDwej8fj8Xg8Ho/H4/F4PB6Px+PxeDwej8fj8Xg8Ho/H4/F4PB6Px+PxeDwej8fj8Xg8Ho/H4/F4PB6Px+PxeDwej+deuiRJOtIEFDzXKG0yD3sAAAAASUVORK5CYII=

}}

{{ iVBORw0KGgoAAAANSUhEUgAAAEQAAAAoAgMAAAA6S7QAAAAACVBMVEUAAAAAAAD///+D3c/SAAAAAXRSTlMAQObYZgAAAKtJREFUKM/FkL0VwyAMhEXBCNqHESgkiozAPh4hRTRl9AM4uLG7XIF53x3yPUGBXQnqhWSgceFhIbB/WRUeA9og5IkSAVZ7ElIzK6EZ8ZAGkGOKS69VX5S8SElOKi5S1VdCCzChk10F0i2BRwTuCA3SDz3aZxBrLG89XmKtgzQxu8sRxF2zxZJjThe1m3gy+tyRvM8Z2/j519zP2WeSs7PveVcF/COh6+KfkS8jFsW42rMUsAAAAABJRU5ErkJggg==

}}

{{ 

}}

{{ 

}}

# Appendix

Honorable Mentions

SWF

This is the infamous Adobe flash viewing file format. Worth mentioning a lot because it allowed for vector layering and could produce a still image that is really similar to the proposed .etxt; however .swf is far too broad and additionally is proprietary and Adobe and other larger tech companies are forcibly stifling the format (ex: when using Google’s advanced search options filetype:swf along with any search query returns zero results). < cite articles about all the tech companies doing that not just google

RAW

I initially thought this was a format but it is a collection of formats designed to be specific to a camera sensor. Due to it being somewhat irrelevant to this project, this entry is brief. Essentially these files are more like the raw ones and zeros coming from the sensor instead of a description of pixels – useful for losslessly preserving photographs.

WEBP

This format, created by Google in 2010, actually covers two different codecs; the lossless and lossy version and both also support being animated. The WEBP format is simply a modern iteration of the techniques that PNG and JPEG combined with the animated property of GIF, which is the only reason the GIF format still sees such high usage. The improvements on standards introduced nearly 30 years ago does have a significant space savings advantage and WEBP is an open format. While I would like to initially support WEBP, the implementation details have even more complicated strategies than JPEG and PNG.[[30]](#footnote-29) As of 2020, Safari 14 was released and marked an end to the last holdout of the highest used web browsers to not support WEBP.

GIF

This is a lossless format that stands for Graphics Interchange Format which was developed by a team of employees by the CompuServ company in 1987. Each pixel can have 8 bits and is limited to a palette of 256 colors from the RGB space. Notably one of the vanishingly few reasons that GIF is still used over better file formats is it being animated without sound. Many chat applications allow the format to play automatically without the end user needing to click or mouse over the file for it to play the animation. This format is technically lossless but can only be used for images with 256 colors, so usually the image has to be compressed before being passed to GIF compression.

GIF’s animation structure is complicated, which is why the engineers behind PNG did not want to immediately make a free version of it. The file has a screen element that makes up the visible area, where one or more images can be viewed through it. GIF supports animation frames that can have independent animation frames, with varying pauses between each frame. Similarly to TIFF, the screen can have images covering it subdivided into areas and change independently on each frame. The images inside the canvas are stored in a linked list such that even if the optional frames past the second were not understood, the renderer could still see some part of the file.

1. <https://www.adobe.io/content/dam/udp/en/open/standards/tiff/TIFF6.pdf> [↑](#footnote-ref-0)
2. <https://www.youtube.com/watch?v=0me3guauqOU> [↑](#footnote-ref-1)
3. <https://yasoob.me/posts/understanding-and-writing-jpeg-decoder-in-python/> [↑](#footnote-ref-2)
4. <https://www.iso.org/standard/29581.html> [↑](#footnote-ref-3)
5. MNG, an animation capable version of PNG was released soon after in 2001 but lacked widespread adoption. [↑](#footnote-ref-4)
6. <https://datatracker.ietf.org/doc/html/rfc2083#section-3> [↑](#footnote-ref-5)
7. <http://edition.cnn.com/TECH/computing/9907/06/1963.idg/> [↑](#footnote-ref-6)
8. Huurdeman, A. A. (2003). *The worldwide history of Telecommunications*. J. Wiley. [↑](#footnote-ref-7)
9. [www.dpbsmith.com/tj2.html](http://www.dpbsmith.com/tj2.html) [↑](#footnote-ref-8)
10. <http://web.mit.edu/Saltzer/www/publications/CC-244.html> [↑](#footnote-ref-9)
11. <https://webarchive.loc.gov/all/20100506231949/http://cm.bell-labs.com/cm/cs/who/dmr/hist.html> [↑](#footnote-ref-10)
12. <https://troff.org/history.html> [↑](#footnote-ref-11)
13. <https://www.historyofinformation.com/detail.php?entryid=3793> [↑](#footnote-ref-12)
14. <http://lamport.azurewebsites.net/pubs/pubs.pdf> [↑](#footnote-ref-13)
15. <https://www.adobe.com/content/dam/acom/en/devnet/actionscript/articles/PLRM.pdf> [↑](#footnote-ref-14)
16. <https://web.archive.org/web/20180727185332/http://www.bitsavers.org/pdf/xerox/xns/standards/XSIG_038306_Introduction_to_Interpress_Jun1983.pdf> [↑](#footnote-ref-15)
17. [https://support.microsoft.com/en-us/office/support-for-eps-images-has-been-turned-off-in-office-a069d664-4bcf-415e-a1b5-cbb0c334a840](https://support.microsoft.com/en-us/office/support-for-eps-images-has-been-turned-off-in-office-a069d664-4bcf-415e-a1b5-cbb0c334a840?ui=en-us&rs=en-us&ad=us) [↑](#footnote-ref-16)
18. <https://www.adobe.com/acrobat/about-adobe-pdf.html> [↑](#footnote-ref-17)
19. <http://www.sgmlsource.com/history/roots.htm> [↑](#footnote-ref-18)
20. <https://www.loc.gov/preservation/digital/formats/fdd/fdd000465.shtml> [↑](#footnote-ref-19)
21. <https://www.w3.org/XML/9712-reports.html> [↑](#footnote-ref-20)
22. <https://www.w3.org/2012/08/history-of-the-web/origins.htm#c4p4> [↑](#footnote-ref-21)
23. <https://techcommunity.microsoft.com/t5/microsoft-365-blog/helping-users-stay-safe-blocking-internet-macros-by-default-in/ba-p/3071805> [↑](#footnote-ref-22)
24. <https://www.wired.com/story/microsoft-disables-macros-default-security-phishing/> [↑](#footnote-ref-23)
25. <http://www.martinreddy.net/gfx/2d/IFF.txt> [↑](#footnote-ref-24)
26. <https://leon.bottou.org/publications/pdf/jei-1998.pdf> [↑](#footnote-ref-25)
27. <https://web.archive.org/web/20090824155127/http://partners.adobe.com/public/developer/en/pdf/PDFReference.pdf> [↑](#footnote-ref-26)
28. <https://web.archive.org/web/20210807021451/https://www.adobe.com/content/dam/acom/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference15_v6.pdf> [↑](#footnote-ref-27)
29. <https://web.archive.org/web/20051103044315/http://www.pdf-tools.com/public/downloads/pdf-reference/pdfreference12.pdf> [↑](#footnote-ref-28)
30. <https://www.youtube.com/watch?v=F1kYBnY6mwg> [↑](#footnote-ref-29)